EVOLVING NEURO-FUZZY TOOLS

FOR

SYSTEM CLASSIFCATION AND PREDICTION

by

Josip Vrbanek Jr.

A thesis submitted

in partial fulfillment of the requirements for the degree of

Master of Science in Control Engineering

LAKEHEAD UNIVERSITY
DEPARTMENT OF ENGINEERING

August, 2008

Your file Votre référence
ISBN: 978-0-494-43436-9
Our file Notre référence
ISBN: 978-0-494-43436-9

Abstract

Classification and prediction algorithims have recently become very powerful tools to a wide array of real-world applications. Some real world applications include system condition monitoring, bioinformatics, robotics, predictive control, earthquake prediction, weather forecasting, stock market and traffic pattern prediction, just to name a few.

Within this work, several novel approaches, as well as modifications to some existing approaches, are introduced in order to improve the performance of current classification and prediction paradigms.

In the first section of this work, a novel weighted recurrent neuro-fuzzy inference system is introduced alongside two existing neural networks. It is found that the novel design outperforms both the existing neural networks in terms of equal-step and sequential-step inputs for time-series forecasting.

The second contribution of this work is the development of a novel evolving clustering algorithim for classification and prediction. This particular algorithim starts without any priori knowledge of the distribution of the data set. The novel design is capable of revealing the true cluster configuration in a single pass of the data, estimating the location and variance of each cluster. After a rigorous performance evaluation, it is found that the novel design outperforms many existing clustering approaches including the well-known potential-based evolving Takagi-Sugeno (eTS) clustering scheme.

The third and fourth contributions of this work are the development of a second novel clustering technique and a novel hybrid training technique.

The clustering technique is a combination of the aforementioned scheme and the potential-based technique. The new training algorithm is a combination of the decoupled-extended Kalman filter (for the backward pass) and the recursive least-sequares estimate (for the forward pass). It is found that the novel clustering technique outperforms many available clustering techniques. Also, the novel training algorithm is proven to outperform most existing training techniques.

Acknowledgment

Completing a MSc. in Engineering is truly a marathon event, and I would not have been able to complete this journey without the aid and support of countless people over the past two years. I must first express my gratitude towards my advisor, Professor Wilson Q. Wang. His leadership, support, attention to detail, hard work, and scholarship have set an example I hope to match some day.

Special thanks go to my two external examiners Dr. Xiaoping Liu and Dr. Carlos Christoffersen; your insights and suggestions have been quite illuminating. An extended special thank you goes to all my professors who taught me the foundational knowledge required to complete my thesis through lectures, discussions and guidance: Dr. Krishnamoorthy Natarajan, Dr. Xiaoping Liu, Dr. Abdelhamid Tayebi and Dr. Wilson Q. Wang.

I'd like to thank the many graduate and undergraduate students I have worked with over the past two years: Peter Liu, Xijiang Dou, Zhirui Huang, Aaron Ji, Aiden Nix, Andrew Roberts and Saleh Ahmad. They each helped make my time in the MSc. program more fun and interesting.

Finally, I thank my parents and friends for instilling in me confidence and a drive for pursuing my MSc. in Engineering.

Contents

List of Figures

Chapter 1

INTRODUCTION

Chapter 2

FIXED STRUCTURES

Chapter 3

MAPPING CONSISTENCY AND CLUSTER COMPATIBILITY

Chapter 4

THE E^{2}TS AND DEKF-RLSE TRAINING TECHNIQUE

List of Tables

Chapter 2

FIXED STRUCTURES

Chapter 3

MAPPING CONSISTENCY AND CLUSTER COMPATIBILITY

Chapter 4

THE E^{2}TS AND DEKF-RLSE TRAINING TECHNIQUE

List of Acronyms

ANFIS:	adaptive neuro-fuzzy inference system
BPTT:	back propagation through time
CBPTT:	constructive back propagation through time
DENFIS:	dynamic evolving neuro-fuzzy inference system
DEKF:	decoupled-extended Kalman filter
E^2TS:	enhanced-evolving Takagi-Sugeno
EBPTT:	exploratory back propagation through time
EFuNN:	evolving fuzzy neural network
EFS:	evolving fuzzy system
ESOM:	evolving self-organizing mapping system
eTS:	evolving Takagi-Sugeno
FDEKF:	fully decoupled extended Kalman filter
FFNN:	feed-forward neural network
FTDNN:	focused time-delay neural network
GA(s):	genetic algorithm(s)
GD:	gradient decent
GEKF:	global extended Kalman filter
MF(s):	membership function(s)
MISO:	multi-input single-output
MLP:	multi-layered perception
MSE:	mean-square error
NARX:	non-linear autoregressive network with exogenous inputs
NF:	neuro-fuzzy
NN:	neural network
NDEKF:	node decoupled extended Kalman filter
RAL:	resource allocation network
RBF:	radial bias functions
RLM:	recursive Levenberg- Marquardt
RN:	recurrent networks
RNN:	recurrent neural networks
RLSE:	recursive least squares estimate
SD:	steepest descent
SOFIS:	self-organizing fuzzy inference system
SONFIN:	self-constructing neuro-fuzzy inference network
TS:	Takagi-Sugeno
TSK:	Takagi-Sugeno-Kang

Chapter 1

INTRODUCTION

This Introduction is organized as follows: Classification, prediction and problems associated with them are defined in Section 1.1. Section 1.2 gives a brief description and some background information on fuzzy logics. In Section 1.3, neural networks are introduced. Section 1.4 introduces the adaptive neuro-fuzzy inference system (ANFIS) which is used as a foundation for most work carried out in the following chapters. In Section 1.5, a brief literature survey is given. And, finally, in Section 1.6, the thesis is outlined.

1.1 Classification and Prediction

Classification is the process by which large amounts of data are divided into different categories, or classes, for evaluation. The Wisconsin Breast Cancer Data [1], for example, consists of nine cell features and 683 patterns to evaluate and classify as either benign or malignant. In order to evaluate the data, one must determine a distinguishable pattern between the input data (the features) and the output data (the classes). The problem of classification arises when there is no distinguishable pattern (or mapping) of the data. It is this problem that fuels the classification genre of research.

Prediction, or time-series forecasting, is to apply the available information of a system to forecast the system's future states. It encompasses the same problems as classification with the exception of organizing the output data into a set of classes. In prediction, features are the inputs as in classification, but the outputs are predicted data.

The use of neural networks and fuzzy logics to solve the aforementioned problem appears to be very promising. When neural networks and fuzzy logic paradigms are trained properly they prove to be a very useful tool for the non-linear mapping that exists in classification and prediction. In order to understand the problem, the next several sections will give a brief introduction to neural networks, fuzzy logics and the adaptive neuro-fuzzy inference system.

1.2 Fuzzy Logics

1.2.1 Introduction to Fuzzy Logics

Fuzzy set theory was first introduced by L.A. Zadeh in his blockbuster paper [2]. Shortly after, fuzzy logic was derived from it for use in many applications. Applications include, but are not limited to, air conditioners, cameras, elevators, pattern recognition and remote sensing.

Fuzzy logic is a form of multi-variable logic derived from fuzzy set theory; it introduces vagueness by eliminating sharp boundaries that divide members from nonmembers in a group. For example, consider the running speed of a vehicle v. Traditional logic would represent v by specific quantities such as 30, 60, and 100 km/h.

In a fuzzy system, however, v can be represented by fuzzy sets such as slow, fast, very fast, which have greater generality, higher expressive power, and enhanced capability to model real-world problems. An example is given in the following section to illustrate fuzzy reasoning processes.

1.2.2 A Fuzzy Reasoning Example

Given a number between 0 and 10 that represents quality of service and cleanliness, what is the right amount of money to tip your housekeeper? The following example will use fuzzy reasoning to determine the best tip.

Step 1: *Linguistically representing the input and output variables.*

We have two input variables; cleanliness and service. Service will be divided into three linguistic terms; poor, good and excellent. Cleanliness, however, will be divided into two terms; dirty or clean. The output variable, tip = [0, 30%], will be represented as cheap, average and generous.

Theoretically, any type of function can be used to represent the membership functions herein. Typical functions include, but are not limited to, the Gaussian bell, triangular, trapezoidal and sigmoidal functions [3]. For our example, we will use Gaussian for the service, trapezoidal for cleanliness and triangular for the tip as shown in Figures 1.1, 1.2 and 1.3, respectively.

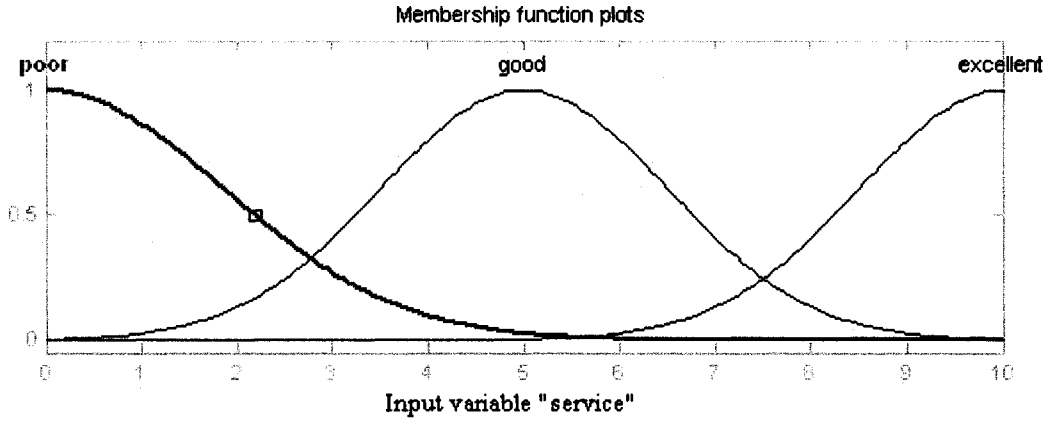

Figure 1.1. Membership functions for service.

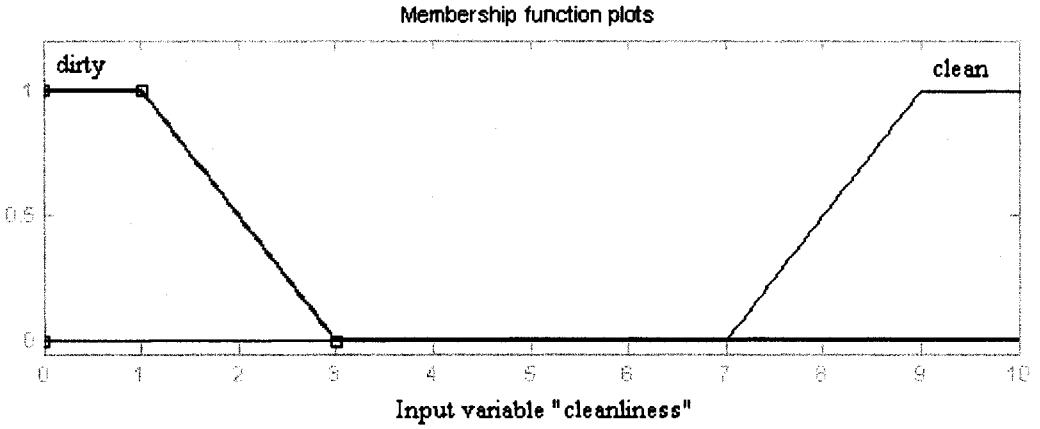

Figure 1.2. Membership functions for cleanliness.

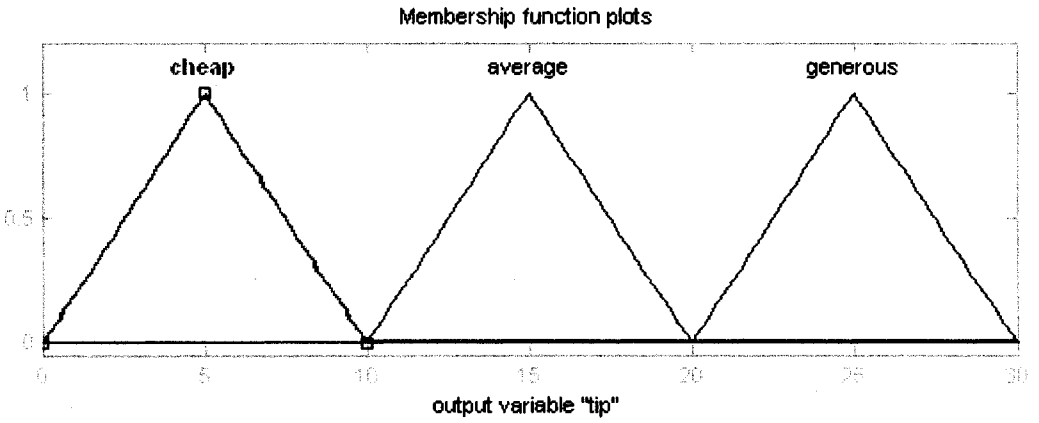

Figure 1.3. Membership functions for tip.

Step 2: *Establishing the rule-base.*

The following rules are established for fuzzy reasoning in this example:

1. *IF the service is poor OR the cleanliness is dirty, THEN the tip is cheap*
2. *IF service is good, THEN tip is average*
3. *IF service is excellent OR cleanliness is clean, THEN tip is generous.*

Step 3: *Fuzzy reasoning.*

When an input is given and the related values are determined from the corresponding Membership Function(s) (MFs), the rule outputs can be computed by appropriate fuzzy operators [3], such as T-norm (AND) and T-conorm (OR) [3]. For our example we will use the highest value for an OR operation and the lowest value for an AND operation.

Step 4: *Defuzzification.*

In order to obtain output information from the fuzzy reasoning system, we must apply an appropriate type of defuzzification technique [3]. For our example, we will use the centriod method.

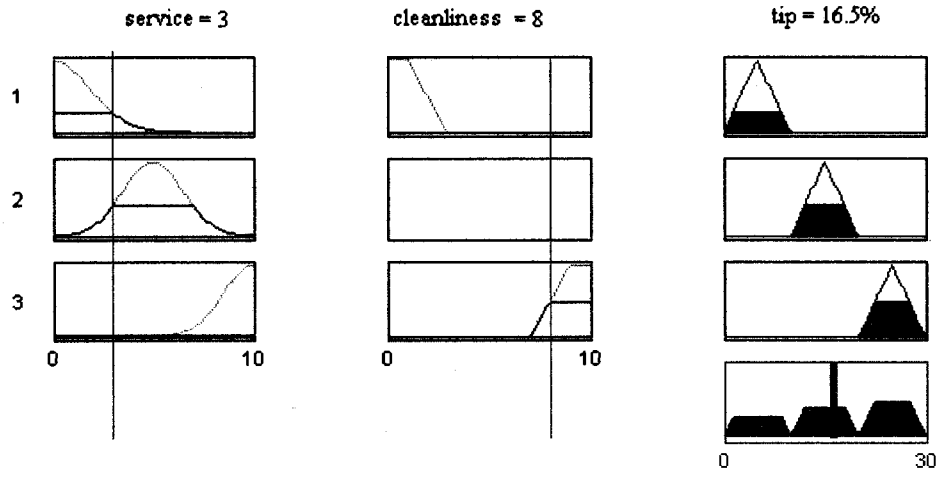

Figure 1.4. Rule-base for the fuzzy tipper.

The finished fuzzy structure is depicted in Figure 1.4. For example, if service was poor (given a 3/10) and cleanliness was clean (given 8/10), the resulting tip is 16.5%.

A majority of the work in this thesis deals with combining fuzzy reasoning with neural network-based training. The next section gives an overview of neural networks followed by an explanation of the adaptive neuro-fuzzy inference system (ANFIS).

1.3 Neural Networks

A Neural Network (NN) is an interconnected group of neurons that use a mathematical model for processing data. NNs are adaptive, or are comprised of equations with adaptive coefficients, that can change its structure, or coefficients, based on input and output data. The adaptive nature of NNs allows for a wide area of applications such as function approximation, regression analysis, time-series prediction and modeling, just to name a few.

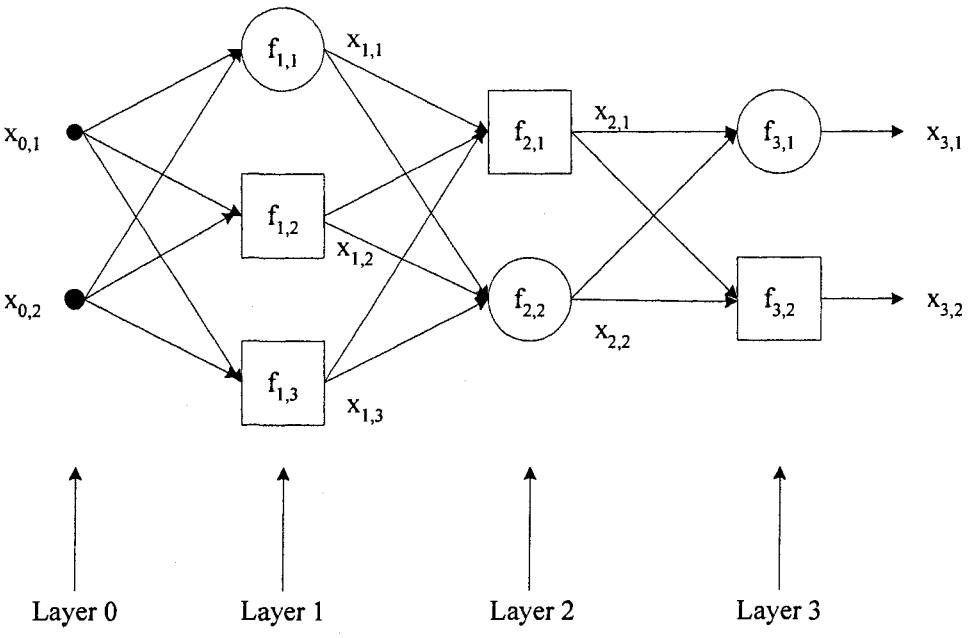

Figure 1.5. A two-input two-output neural network.

NNs can have as many inputs, outputs, layers, recursive links, or functions as required by the designer, but for simplicity, we will examine a two-input two-output feed-forward NN as shown in Figure 1.5. $f_{L,k}$ represents a function where L is the layer and k is the component of the actual output vector produced by P entries. $f_{L,k}$ can take on many different functions such as hard-limit, linear and log-sigmoid functions [24]; depending on specific mapping applications.

NNs act as mapping approximators between input and output data spaces. Through a series of iterations, the NNs can be trained by an appropriate algorithm which modifies the network coefficients so as to optimize the mapping between the input and output data. Since a major part of this thesis deals with novel and existing training techniques, several training algorithms will be discussed in full detail in the following chapters.

1.4 The ANFIS structure

The Adaptive Neuro-Fuzzy Inference System (ANFIS), in fact, is a fuzzy reasoning scheme whose system parameters are trained by using NN-based algorithms. Consider n input variables, $\{x_1, x_2, \cdots, x_n\}$, which are normalized in [0, 1], the fuzzy reasoning is performed by the following general representation

$$\Re_j: \text{IF } (x_1 \text{ is } A_1^j) \text{ AND } (x_2 \text{ is } A_2^j) \text{ AND } \ldots (x_n \text{ is } A_n^j) \text{ THEN } (y_j = f_l^j) \quad (1.1)$$

where $\Re_j$ denotes the jth fuzzy rule, $j \in [1, N]$; N is the total number of fuzzy rules (clusters); A_i^j is the jth fuzzy set for x_i, $i \in [1, n]$; $y_j = [y_{j1}, y_{j2}, \cdots, y_{jM}]$ is an M-dimensional consequent (output) structure.

Based on the consequent reasoning structure, three types of reasoning paradigms are commonly used in fuzzy applications:

Type I (Mamdani model): $f_l^j = D_l^j$, where D_l^j is a consequent fuzzy set,

Type II (zero order TS model): $f_l^j = a_l^j$, where a_l^j is a singleton*,

Type III (first order TS model): $f_l^j = b_{0l}^j + b_{1l}^j x_1 + \cdots + b_{nl}^j x_n$,

where $b_\bullet^j$ are consequent parameters; $j \in [1, N_r]$, N_r is the total number of fuzzy rules and $l \in [1, M]$, M is the total number of outputs.

Type I models are mostly used in diagnostics and control applications due to their properties of diversified selection in the shape of membership functions, fuzzy reasoning, and defuzzification methods [4]. A Type II model, however, is simply a special case of the Type I paradigm in which the consequent term becomes a singleton. A Type III model is mostly used on higher order or nonlinear systems due its higher degree of flexibility in modeling. For this thesis, the Type III model will be used exclusively due to the fact that it is more accurate for non-linear mapping. It is worth noting, however, the Mamdani or the zero order TS model can be used in substitute.

Consider a simple example for a two-input first-order Sugeno fuzzy model:

Rule 1: IF x is A_1 **AND** y is B_1 **THEN** $f_1 = p_1 x + q_1 y + r_1$,

Rule 2: IF x is A_2 **AND** y is B_2 **THEN** $f_2 = p_2 x + q_2 y + r_2$,

where the coefficients in f_1 and f_2 are called the consequent parameters and are optimized in the training process. Figure 1.6 depicts the corresponding network architecture.

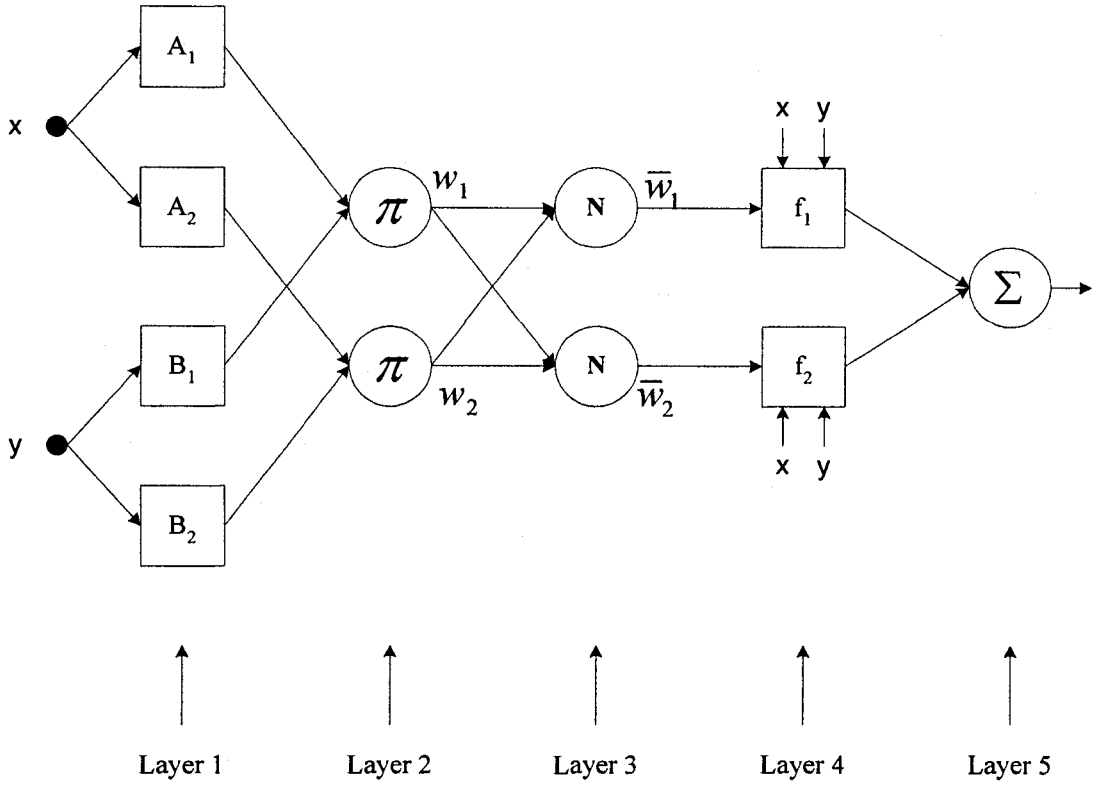

Figure 1.6. A two-input first-order Sugeno fuzzy model.

The input state variables are fuzzified in *layer 1*. The layer output can be represented as:

$$\begin{aligned} O_{1,i} &= \mu_{Ai}(x) \qquad for\, i = 1,2 \quad and \\ O_{1,i} &= \mu_{B(i-2)}(y) \qquad for\, i = 3,4 \end{aligned} \tag{1.2}$$

where x and y are inputs to *layer 1*, and $\mu_{Ai,Bi}$ are the desired membership functions for x and y, respectively.

*A singleton is a fuzzy set whose support is a single point in the universe with a membership function of one.

Fuzzy operation is conducted in l*ayer 2*. The rule output (or firing strength) can be represented as:

$$O_{2,i} = w_i = \mu_{Ai}(x)\mu_{Bi}(y), \quad i = 1,2. \tag{1.3}$$

All the rule outputs (or firing strengths) are normalized *layer 3*, or:

$$O_{3,i} = \overline{w}_i = \frac{w_i}{w_1 + w_2}, \quad i = 1,2. \tag{1.4}$$

The nodes in *layer 4* serve as linear combination of the input variables with additive functions. The outputs are given as:

$$O_{4,i} = \overline{w}_i f_i = \overline{w}_i \left(p_i x + q_i y + r_i\right), \quad i = 1,2. \tag{1.5}$$

Layer 5 conducts defuzzification, and the overall system output becomes

$$O_{5,i} = \sum_i \overline{w}_i f_i = \frac{\sum_i w_i f_i}{\sum_i w_i} \tag{1.6}$$

In current literature, the ANFIS structure, or a modification of it, is widely used to solve the problems associated with classification and prediction. Many authors have used this structure as a foundation for newer and improved paradigms. In the next section, a brief literature review of the current classification and prediction techniques are given.

1.5 Review of the Literature

The review of literature began with a review of neuro-fuzzy (NF) and soft computing techniques. The book authored by Jang *et al.* introduces fuzzy logic and NNs [3]. Some of the NNs reviewed include the feed-forward neural network (FFNN), the focused-time delay neural network (FTDNN) and the non-linear autoregressive network with exogenous inputs (NARX). They also introduce training techniques such as least-squares

estimate, (for the consequent parameters in the Type III model), derivative-based optimization techniques (for the premise parameters) and Genetic Algorithms (GA). The authors also present the ANFIS network and its applications to system control.

The book proved to be an excellent source for reference and foundational knowledge. In addition, the User's Manual for Neural Network and Fuzzy Logic Toolboxes by Mathworks Inc. [24] also provided useful information regarding theoretical review of these reasoning schemes. These toolboxes continue the work from Jang *et al.* and introduce Radial Basis Networks (RBN), adaptive filters, and some advanced topics and training algorithms.

The authors' research group has developed several types of NF classifiers and predictors for machinery condition diagnostics and prognostics [22, 23, 50, 51]. Their investigation results have shown that if an NF scheme is properly trained, it outperforms those based on the feedforward NNs, the recurrent NNs, and fuzzy logic in both classification and forecasting applications.

Reference [5] discussed the issues of multi-step-ahead prediction with NNs. The authors begin with arguing between some of the benefits of global modeling versus local modeling. Local modeling tends to generate too many segments and trainable parameters. Global modeling, through the use of multi-layered perceptron(s) (MLPs) or recurrent network(s) (RNs), for example, builds a single complex model for the entire range of behaviors identified in the time series. On major advantage to global modeling is the decrease in trainable parameters resulting in faster training. Because of the advantage of global modeling, an RN for multi-step-ahead prediction is presented as opposed to single step prediction.

In order to predict more time steps, however, the authors investigate three techniques for the structure training in multi-step ahead prediction: Back Propagation Through Time (BPTT), Constructive Back Propagation Through Time (CBPTT) and Exploratory Back Propagation Through Time (EBPTT). Furthermore, the authors use the well known Mackey-Glass [6] benchmark data for simulation and verification. Experimental results indicate that the EBPTT training technique outperformed the other training techniques up to 14 steps-ahead. Results also indicate that the error of the EBPTT training technique converges faster and to a lower value.

The biggest problem with the aforementioned techniques for prediction (and classification) is the required prior knowledge in selecting the ideal number of rules or number of nodes required for a certain task. Hence, the review of literature continued with evolving rule-base structures.

In evolving rule-base structures the model continually evolves by adding or subtracting rules with more summarization power and by modifying existing rules and parameters. Plaman P. Angelov can be considered the pioneer researcher of evolving rule-base structures. In his paper entitled *Evolving Rule-based Models: A Tool for Intelligent Adaptation* [7], Angelov *et al.* introduce a novel technique for updating or modifying an existing rule-base. As the input/output data iterates, the presented algorithm calculates the potential of the new data point as a rule (or cluster) for the rule-base. If the potential of the new data point is greater than all the existing clusters, a new rule is formed with the current data points and a fixed spread for the cluster radius. After this decision making process, the consequent parameters are trained, online, via recursive least-squares estimate (RLSE).

In reference [8], Angelov *et al.* extend the work in [7] and apply the evolving rule-base technique to a Takagi-Sugeno structure. Takagi-Sugeno structures ensure that the rules are more gradual (that they are able to describe a larger number of data samples) from time of their initiation [8]. The authors were successful in applying their algorithm to air-conditioning installation surveying a real building. Despite the promising results, one problem still loomed in their algorithms. When new rules were generated, the spreads of the newly formed clusters would stay fixed; as well as the location of the clusters. A new training technique that embodies training the premise parameters as well as the consequent parameters would result into a more accurate paradigm.

Angelov *et al.* introduced a technique to recursively change the size of the spreads of all the rules in [9]. Even though this new algorithm did outperform the latter technique, it did not solve the problem of training the location of the clusters.

Wang *et al.* proposed a novel technique for structure identification and simulation [10]. The paper introduced an MCA clustering algorithm that can identify a parsimonious internal structure in the sense that the number of clusters is equal to the true number of clusters; furthermore, a fast recursive linear/non-linear least-squares optimization algorithm is used to accelerate the learning convergence. The novel MCA clustering algorithm outperformed several existing algorithms [11-14] but was not compared to Angelov's latter algorithm.

1.6 Thesis Organization

The objective of this research is to develop advanced intelligent tools with novel structures and novel training techniques to overcome the obstacles of current clustering and prediction techniques. The purpose is to provide a wide array of industries more reliable and robust classifiers and predictors for dynamic system classification and forecasting applications.

In Chapter 2, feed-forward and recurrent NNs are introduced and simulated in order to compare sequential-based inputs to step-inputs. Also in Chapter 2, a novel weighted recurrent NF paradigm is proposed and proven to be a more reliable and robust multi-step ahead predictor.

Chapter 3 deals with self-organizing or evolving NF paradigms for classification and prediction. Chapter 3 begins with introducing some well known techniques and some novel techniques. Also in Chapter 3, a novel clustering algorithm and hybrid training technique is introduced and compared to the literature.

Chapter 4 introduces another novel clustering algorithm and novel training algorithm. The new clustering approach is an enhanced version of the evolving Takagi-Sugeno paradigm suggested by Angelov in [8]. The new training technique is a hybrid technique consisting of the decoupled-extended Kalman filter and the recursive least-squares training technique.

Chapter 5 concludes the thesis and introduces some future work to be carried out.

Chapter 2

NF FORECASTING SCHEMES WITH FIXED STRUCTURES

2.1 Introduction

Multi-step ahead forecasting tools are very useful to a wide assortment of real-world applications. In economical applications, for example, forecasting tools can be used to predict the stock market, the Canadian dollar or the price of oil. In industrial applications, however, forecasted information can be used to schedule repairs and maintenance for important fabrication facilities so as to prevent production performance degradation, malfunction or even catastrophic failures.

Several techniques have been proposed in current literature for time-series prediction [15]. The classical forecasting techniques are mainly based on classical stochastic models which can be derived mathematically. However, it is hard to accurately derive an analytical model for complex systems [16, 17].

Within the last decade, a shift has been made from the classical approaches to the use of knowledge-based data-driven paradigms, such as various neural networks (NNs) [18 - 20] and neuro-fuzzy (NF) paradigms [21 - 23]. Despite the data-driven paradigms being able to demonstrate their superiority in many applications, the paradigms are mainly used for one step-ahead prediction.

It is apparent that advanced research needs to be done before a more robust multi-step-ahead predictor can be used. Accordingly, the objective of this chapter is to 1) compare step inputs to sequential inputs for multi-step ahead prediction and to 2) improve the performance of multi-step prediction by adopting a novel recurrent weighted NF paradigm.

2.2 Feed-Forward Neural Networks

In this section, a feed-forward NN is used to facilitate the idea that a step-input based multi-step predictor outperforms a sequential-step multi-step predictor. In the next subsection the feed-forward structure is introduced. In subsection 2.2.2 the steepest-descent training algorithm is introduced. Simulation results are given in section 2.2.3.

2.2.1 Feed-forward Structure

The network architecture for the step-input based feed-forward neural network is given in Figure 2.1. The variable s represents the time step. For a sequential input structure, the input variables would take on $\{x_0 \quad x_{-1} \quad \ldots \quad x_{-n}\}$ as opposed to $\{x_0 \quad x_{-s} \quad \ldots \quad x_{-ns}\}$ for step-based inputs. The latter subscript represents the s time step whereas the former represents the numbered input.

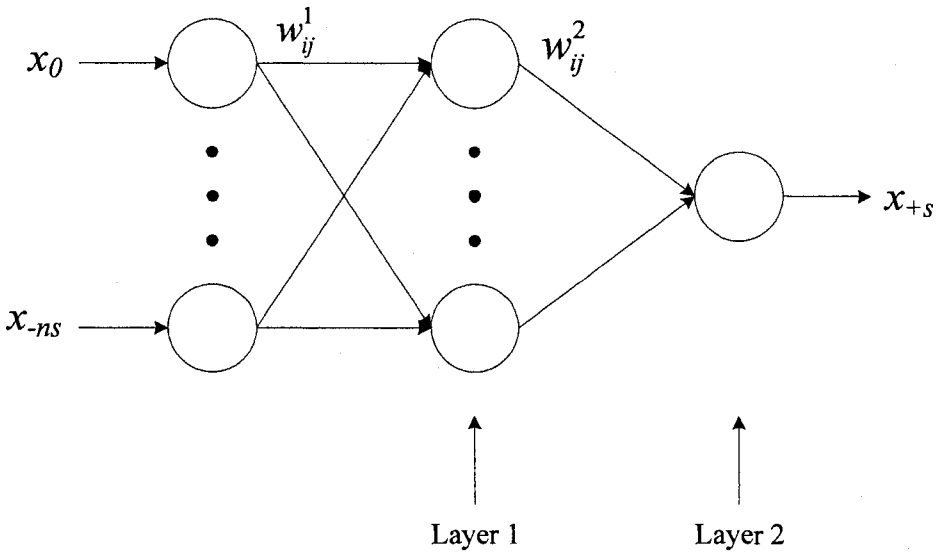

Figure 2.1. Network architecture for the step-input based feed-forward NN.

The nodes in *layer 1* represent the tan-sigmoidal transfer function [24]. By tests, it is found that the tan-sigmoidal transfer function allows for better input/output mapping for non-linear data in this case.

To get an idea of the number of parameters to be trained, consider $n = 9$. In this situation, ten nodes will be used in *layer 1* and 1 node will be used in *layer 2* with a total of 11 biases. The total number of parameters to be trained, then, is 121.

2.2.2 The Steepest-Descent Algorithm

There exist several training techniques for the aforementioned system parameters [3]. This section introduces one well known training technique called the Steepest-Descent (SD), or Gradient-Descent (GD), algorithm [3]. This method is one of the classical optimization techniques for minimizing a given function defined on a multidimensional input space [3].

The gradient of a differentiable function E, or objective function, as in this case, is the vector of first derivatives of E, denoted as $\boldsymbol{g}$. Hence,

$$\vec{g} = \left[\frac{dE(\vec{\theta})}{d\theta_1}, \quad \frac{dE(\vec{\theta})}{d\theta_2}, \quad \cdots \quad \frac{dE(\vec{\theta})}{d\theta_n} \right] \tag{2.1}$$

where θ_i is an adjusTable (trainable) parameter for $i = 1,...,n$ parameters. The objective function E is a user defined function to be minimized and, in this case, is denoted as

$$E(\vec{\theta}) = \frac{1}{2} \sum_{k=0}^{P-1} \left[x_{+s}(\vec{\theta}_k) - d_k \right]^2 \tag{2.2}$$

where d_k represents the desired value, $x_{+s}(\vec{\theta}_k)$ represents the system output as a function of the adjustable parameters at time step k and P is the total number of training data pairs.

With the above, the steepest descent algorithm can be defined as:

$$\vec{\theta}_{k+1} = \vec{\theta}_k + \eta \vec{g} \tag{2.3}$$

where η is the learning rate and is calculated with line search techniques [10].

The training technique discussed in this section, including all the corresponding equations, are applied to the feedforward NN, recurrent NN and the novel NF paradigm in the following sections and subsections of this chapter.

2.2.3 Experimental Results

Experiments are performed by a series of simulation tests based on data sets from the Mackey-Glass equation [6],

$$\frac{dx(t)}{dx} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t). \tag{2.4}$$

The Mackey-Glass equation has specific natures such as chaotic, non-periodic and non-convergence and has been used as a benchmark for evaluating the performance of predictors. Initial conditions used for the simulations are $\tau = 30$, $x(0) = 1.2$, $dt = 1$, and $x(t) = 0$ for $t < 0$.

For this experiment, the FFNN will have 10 inputs, 12 nodes in the hidden (second) layer and one output node. The total number of parameters to be updated is 155, that is, 120 input weights, 12 output weights, and 23 biases.

The FFNN is tested with 10 inputs incorporating time steps s = [3, 6, 9 and 12]. For each time step, the system is trained over 100 epochs[†] and 800 data points from the equation (2.4). Each test is repeated over 10 times to minimize random effects related to initial values of weights. After the training period, 400 data pairs are used for verification to prevent over training.

Figure 2.2 depicts the results for the sequential based FFNN with 10 inputs and a 12 step-ahead prediction capability. Figure 2.3 illustrates the results for the step-input based FFNN with 10 inputs and a 12 step-ahead prediction capability. Table 2.1 summarizes the mean-square errors (MSE) and convergence errors from training.

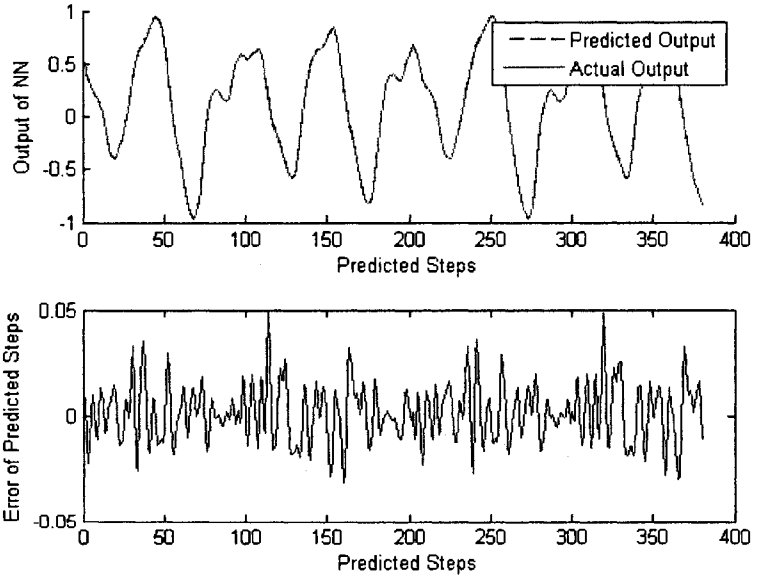

Figure 2.2. 12 step prediction via FFNN by means of sequential input steps.

[†] An epoch refers to the entire range of given data sets.

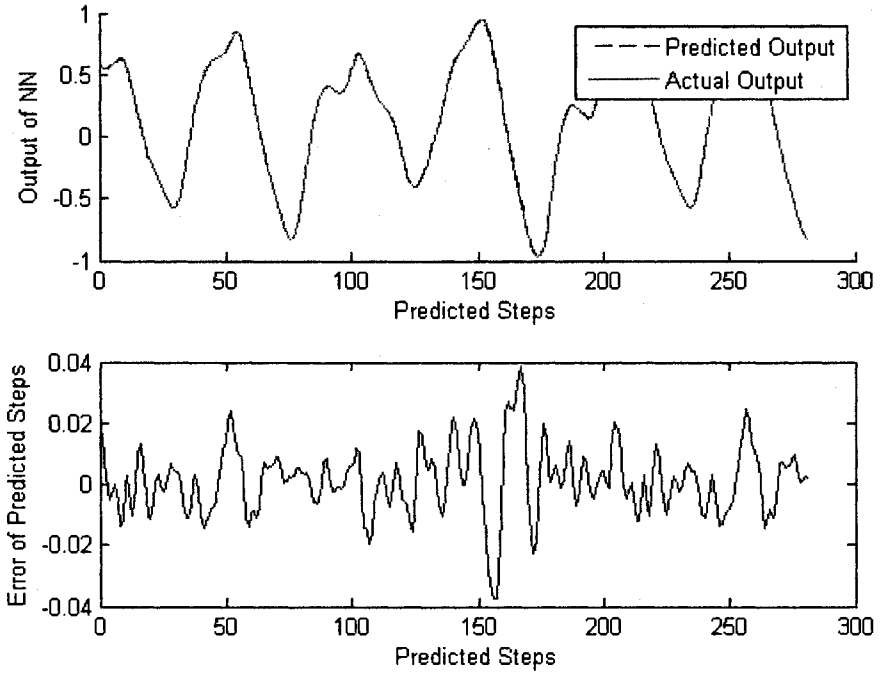

Figure 2.3. 12 step prediction via FFNN by means of equal input steps.

Table 2.1. Results for the sequential and step based FFNNs.

	Sequential				Equal step			
Epoch	3	6	9	12	3	6	9	12
0	4.26E+00	5.79E-01	3.99E+00	1.23E+00	6.52E+00	3.37E-01	2.33E+00	2.69E+00
10	6.70E-04	3.07E-03	1.79E-03	1.27E-03	1.75E-05	2.73E-05	1.55E-03	2.04E-03
20	6.13E-04	2.40E-03	1.53E-03	9.89E-04	6.58E-06	7.46E-06	2.21E-05	4.20E-04
30	5.44E-04	2.16E-03	1.49E-03	8.56E-04	3.56E-06	5.90E-06	1.66E-05	2.70E-04
40	5.18E-04	2.08E-03	1.45E-03	7.90E-04	2.66E-06	4.90E-06	1.36E-05	2.15E-04
50	4.90E-04	1.91E-03	1.42E-03	7.00E-04	2.16E-06	4.38E-06	1.23E-05	1.56E-04
60	4.59E-04	1.85E-03	1.40E-03	7.00E-04	1.84E-06	3.92E-06	1.21E-05	1.23E-04
70	4.37E-04	1.68E-03	1.35E-03	7.00E-04	1.44E-06	3.49E-06	1.19E-05	5.88E-05
80	4.26E-04	1.62E-03	1.31E-03	7.00E-04	1.02E-06	3.08E-06	1.17E-05	4.86E-05
90	4.19E-04	1.56E-03	1.28E-03	7.00E-04	8.68E-07	2.70E-06	1.15E-05	4.43E-05
100	4.10E-04	1.50E-03	1.26E-03	6.90E-04	8.13E-07	2.55E-06	1.12E-05	3.82E-05
MSE	9.30E-05	3.62E-04	2.55E-04	1.77E-04	7.44E-07	9.48E-06	8.44E-06	1.46E-04

Table 2.1 shows that the equal step predictor outperforms the sequential step predictor in training and verification.

For both structures the training errors converge, although, for the equal step predictor, the errors converge to a smaller number incorporating the same number of epochs as the sequential step predictor. Also, Table 2.1 verifies the fact that it becomes increasing difficult to train predictor systems requiring higher prediction steps.

In the following section, the recurrent NN is introduced and compared to the feed-forward NN.

2.3 Recurrent Neural Network

In this section, a recurrent NN is used to enhance the idea that a step-input based predictor outperforms a sequential-step multi-step predictor. Also in this section, the recurrent NN is compared to the feed-forward NN.

In the following subsection, the recurrent structure is introduced. Experimental results are given in Section 2.3.2.

2.3.1 The Recurrent Structure

The network architecture for the step-input based recurrent NN is given in Figure 2.4. The architecture resembles the FFNN with the exception of the recurrent links. The recurrent links provide a better mapping of the input/output data because historical information is utilized in the system.

As in the case of the FFNN, the variable s represents the time step. For a sequential input structure, the input variables would take on $\{x_0 \quad x_{-1} \quad \ldots \quad x_{-n}\}$ as opposed to $\{x_0 \quad x_{-s} \quad \ldots \quad x_{-ns}\}$ for step-based inputs.

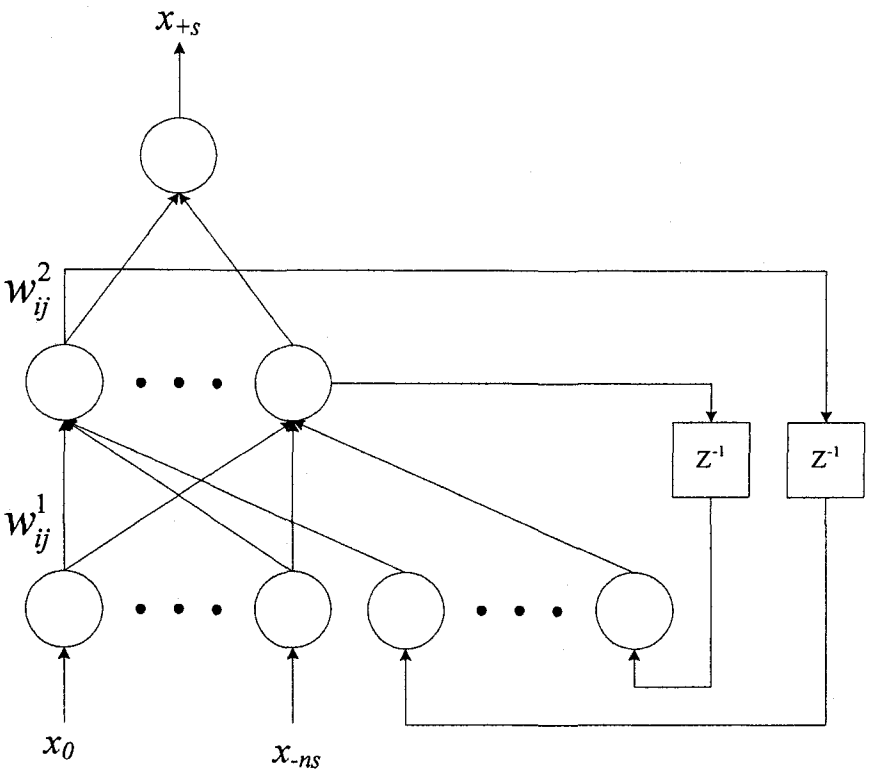

Figure 2.4. The RNN.

The nodes in *layer 2* represent a special case of the tan-sigmoidal transfer function [24]. The input to the tan-sigmoidal transfer function is a combination of the input variable x and a single step delay of the output of the corresponding tan-sigmoidal function. This configuration will prove to be a more accurate model for multi-step-ahead prediction.

2.3.2 Simulation Results

The performance of a knowledge-based data-driven forecasting scheme is directly related to its reasoning structure and the number of network parameters [25]. To facilitate the comparison study among the FFNN, the number of network parameters will be kept comparable.

Consider (n+1) input state variables $\{x_0\ x_{-s}\ \cdots\ x_{-ns}\}$, which represent the current (x_0) and the previous states of a dynamic system with a time step s. If ten inputs are employed in the input layer, that is, n = 9, ten nodes will be used in the recurrent context layer. The s-step-ahead state, x_{+s}, will be given from the output node. In total, there are 151 parameters to be updated in this predictor: 100 input weights (w_{ij}^1), 10 output weights (w_{ij}^2), 10 recurrent weights (w_{ij}^3), and 31 biases; a similar number of parameters to the aforementioned FFNN.

Figure 2.5 depicts the results for the sequential based RNN with 10 inputs and a 12 step-ahead prediction capability. The error in Figure 2.5 converges after approximately 25 samples. This is due to the fact that a unit step delay is used for predicting s = *12* time steps. Figure 2.6 illustrates the results for the step-input based RNN with 10 inputs and a 12 step-ahead prediction capability. Table 2.2 summarizes the verification errors (in MSE) and convergence errors from training.

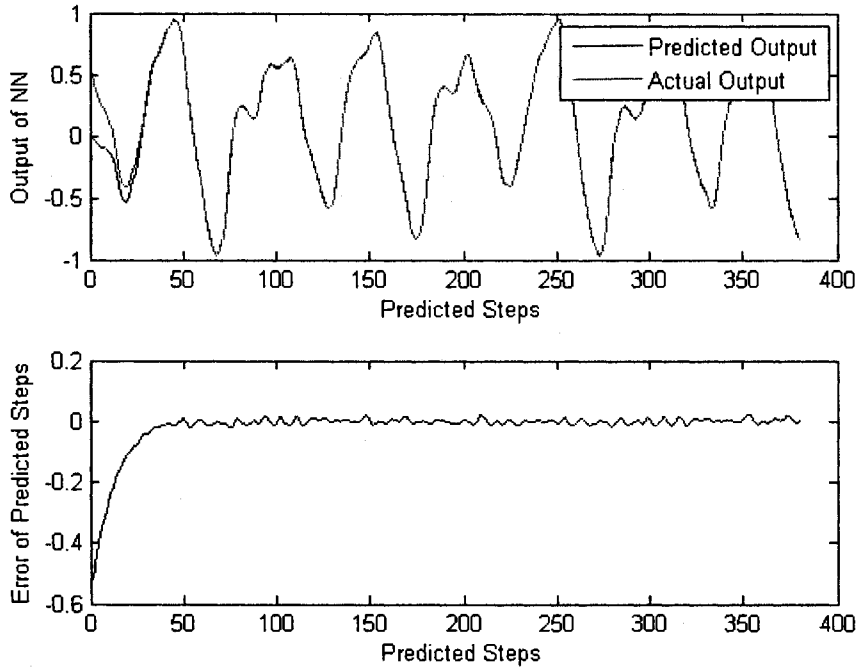

Figure 2.5. 12 step prediction via RNN by means of sequential input steps.

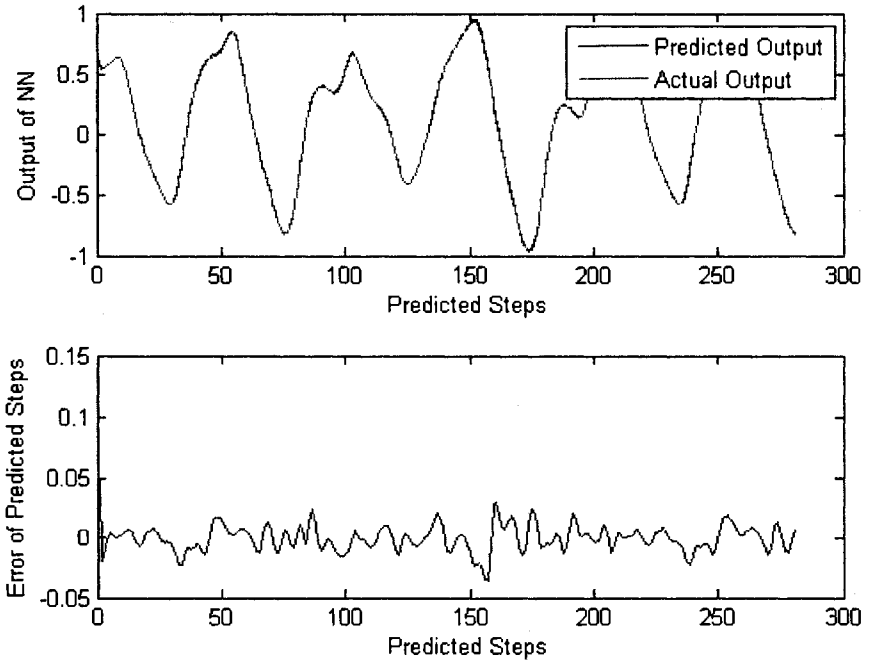

Figure 2.6. 12 step prediction via RNN by means of equal input steps.

Table 2.2. Results for the sequential and equal-step RNN.

	Sequential				Equal step			
Epoch	3	6	9	12	3	6	9	12
0	1.36E+01	2.25E+00	5.07E+00	1.11E+00	7.41E-01	3.49E+00	5.38E-01	3.24E-01
10	2.82E-03	3.80E-03	4.83E-03	1.27E-03	1.65E-06	7.67E-04	4.51E-04	6.30E-04
20	4.90E-04	2.14E-03	1.64E-03	5.01E-04	1.39E-06	2.20E-05	3.73E-05	2.12E-04
30	2.46E-04	1.70E-03	1.40E-03	3.47E-04	1.23E-06	1.21E-05	2.73E-05	1.00E-04
40	1.61E-04	1.70E-03	1.35E-03	2.91E-04	1.15E-06	7.02E-06	2.11E-05	7.20E-05
50	1.43E-04	1.66E-03	1.27E-03	2.55E-04	9.17E-07	5.09E-06	1.48E-05	6.21E-05
60	1.34E-04	1.50E-03	1.24E-03	2.22E-04	8.77E-07	4.34E-06	1.20E-05	5.54E-05
70	1.27E-04	1.29E-03	1.10E-03	1.93E-04	8.32E-07	3.87E-06	1.00E-05	4.98E-05
80	1.00E-04	1.18E-03	1.10E-03	1.61E-04	7.84E-07	3.53E-06	8.36E-06	4.49E-05
90	1.00E-04	9.10E-04	1.10E-03	1.33E-04	7.34E-07	3.29E-06	6.41E-06	4.04E-05
100	1.00E-04	7.90E-04	1.10E-03	1.15E-04	6.85E-07	3.11E-06	6.22E-06	3.65E-05
MSE	5.40E-04	1.33E-04	1.02E-04	9.90E-05	3.58E-07	5.89E-06	3.99E-06	8.88E-05

Simulation results of the RNN are analogous to that of the FFNN. Table 2.2 shows that the equal step predictor outperforms the sequential step predictor in training and verification. For both sets of inputs, the training errors converge, although, for the equal step predictor, the errors converge to a smaller number incorporating the same number of epochs as the sequential step predictor. Also, as in the case of the FFNN, Table 2.2 verifies the fact that it becomes increasing difficult to train structures of higher step-inputs.

When comparing the mean squared errors for the RNN and the FFNN, it can be seen that the errors are significantly lower for the RNN for all time steps. This is proof that the step-input based recurrent predictor outperforms all other predictors mentioned so far in this chapter. Results of the above simulation have been published in the Journal of Computer Science [25].

Following the results obtained in the previous sections, the next section will compare a NF inference system to a novel recurrent NF inference system. The novel design, which will incorporate step-inputs and recurrent links, will prove to be a more reliable and robust multi-step predictor.

2.4 The Step-input-based recurrent weighted NF Predictor

In this section, a novel weighted recurrent NF predictor is developed and compared to the other predictors introduced in this chapter. The idea of the proposed novel design is to create a more reliable and robust predictor for industrial applications. In the next subsection, the novel design is introduced followed by an introduction to a hybrid training technique. This section concludes with a simulation and comparison in subsection 2.4.3.

2.4.1 The Recurrent Structure

An NF reasoning scheme applies a set of fuzzy IF-THEN rules to describe the input/output data mapping and forecast the future states of a dynamic system. The fuzzy system parameters are optimized by a hybrid training algorithm. According to some advanced investigation [22], it is found that the first order TS fuzzy paradigm is more flexible in modeling higher order nonlinear systems, and will be adopted in this thesis. To simplify forecasting reasoning, two membership functions (MFs), *small* and *large*, are assigned to each input state variable. The *s*-step-ahead state of the dynamic system x_{+s} can be formulated by:

$\Re_j$: **IF** $(x_0.is.A_1^j)$ and $(x_{-s}.is.A_2^j)\ldots\ (x_{-ns}.is.A_m^j)$

$$\textbf{THEN}\ x_{+s} = c_0^j x_0 + c_1^j x_{-s} + \ldots + c_n^j x_{-ns} + c_{n+1}^j \tag{2.5}$$

where $\Re_j$ denotes the j^{th} fuzzy rule, $j = 1, 2, \ldots, M$, M is the total number of fuzzy rules; $x = \{x_0 \quad x_{-1} \quad \ldots \quad x_{-n}\}$ for sequential inputs; A_k^j is the j^{th} fuzzy set for x_{-is}, $i = 0, 1, \ldots, n$, $k = 1, 2, \ldots, m$, where $m = 2n$ in this case.

To make the network comparable with the previously mentioned NN predictors, four input state variables are applied in this case, that is, $i = 0, 1, \cdots, 3$, $M = 16$, and $m = 2n = 8$, and the number of parameters to be updated is 104.

The network architecture of this weighted recurrent NF predictor is architecturally shown in Figure 2.7 It is a 5-layer network in which each node performs a particular activation function on the incoming signals. The links have unity weights unless specified. Layer 1 is the input layer. Each node in *layer 2* acts as a MF. Different from the general NF schemes [26] and the predictor as proposed in [22], this recurrent NF system has a weighted feedback link to each node in *layer 2*. The recurrent context units copy the activations of output nodes from the previous time steps, and allow the network to remember information from the past.

Given an MF, the actual input of the node at the t^{th} time instant is

$$X_{-is}^{(t)} = x_{-is}^{(t)} + w_{ik} A_k^j\left(x_{-is}^{(t-1)}\right) \tag{2.6}$$

where $x_{-is}^{(t)}$ and $x_{-is}^{(t-1)}$ are, respectively, the input x_{-is} at the t^{th} and $(t\text{-}1)^{th}$ time instants;

$A_k^j\left(x_{-is}^{(t-1)}\right)$ is the node output (membership grade) in the last time step, and w_{ik} is the weight of the feedback link; $i = 0, 1, \ldots, n$; $k = 1, 2, \ldots, m$; and $t = 0, 1, \ldots, P\text{-}1$, where P is the total number of time instants (or training data sets) of interest.

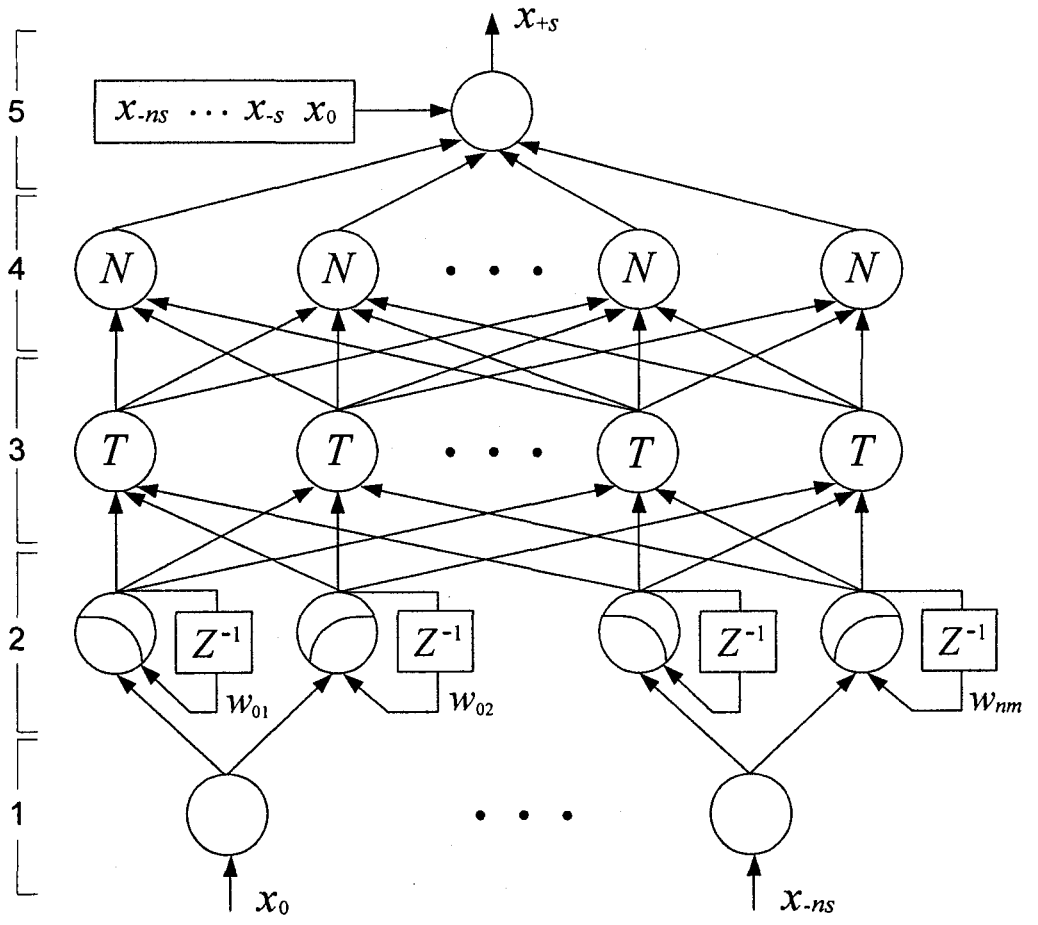

Figure 2.7. Network architecture of the weighted recurrent NF predictor.

Each node in *layer 3* performs a fuzzy T-norm [3] operation. If a max-product operator is applied [3], as in this case, the rule firing strength is

$$\mu_j = \prod_{i=0}^{n} A_k^j\left(X_{-is}^{(t)}\right), \quad j = 1, 2, \ldots, M; \quad i = 0, 1, \ldots, n. \tag{2.7}$$

The rule firing strengths are normalized in *layer 4*. After a linear combination of the input variables in *layer 5*, the predicted output x_{+s}, after s-steps, is computed by using centroid defuzzification [3]

$$x_{+s} = \sum_{j=1}^{M} \bar{\mu}_j (c_0^j x_0 + c_1^j x_{-s} + \cdots + c_n^j x_{-ns} + c_{n+1}^j) \ , \tag{2.8}$$

where $\bar{\mu}_j = \frac{\mu_j}{\sum_{j=1}^{M} \mu_j}$ is the normalized firing strength of the j^{th} rule. The fuzzy system parameters w_{ik} and c_i^j and A_k^j are optimized by the related training algorithms as discussed in the next subsection.

2.4.2 The Hybrid Training Technique

Once the NF predictor structure is created, a forecasting paradigm should be properly trained to optimize the input/output mapping [27-29]. Since the suggested recurrent NF scheme in (2.5) contains both linear consequent weights and nonlinear premise parameters, to improve the training convergence, a hybrid training technique is suggested in this subsection: the nonlinear fuzzy parameters in the recurrent context layer are optimized by using the classical gradient-descent algorithm [3], whereas the consequent linear parameters are fine-tuned by employing a weighted least squares estimate (LSE). A hybrid training technique is superior to classical single training algorithms, because it possesses randomness that may help to escape from local minima and it is also necessary to accommodate different characteristics in time-varying systems.

Consider the t^{th} input data pair $\{x_0^{(t)} \ x_{-s}^{(t)} \cdots x_{-ns}^{(t)}\}$, t = 0, 1, ..., P-1, P is the total number of training data pairs. The forecast state after s steps, $x_{+s}^{(t)}$, is computed by (2.8).

If $d^{(t)}$ denotes the desired output state, which represents the actually observed output value for the $(t+1)^{\text{th}}$ data set, that is, $d^{(t)} = x_0^{(t+1)}$, the forecasting error is defined as

$$E_2 = \sum_{t=0}^{P-1} E_{2t} = \frac{1}{2} \sum_{t=0}^{P-1} \lambda (x_{+s}^{(t)} - d^{(t)})^2 \tag{2.9}$$

where $\lambda \in [0.95 \;\; 1]$ is a weight factor to highlight the contribution of the recent data sets from a time-variant system.

Given the values of the MF parameters and P training data pairs to the adaptive predictor, $\{x_0^{(t)} \;\; x_{-s}^{(t)} \cdots x_{-ns}^{(t)} \;\; d^{(t)}\}$, $k = 0, 1, 2, \ldots, P\text{-}1, P$ linear equations in terms of the consequent parameters $\boldsymbol{\theta}$ can be formed as [25]:

$$\mathbf{R}\boldsymbol{\theta} = \mathbf{d}, \tag{2.10}$$

where $\boldsymbol{R}$ is the resulting matrix from the inference operation of the NF predictor

$$\mathbf{R} = \begin{bmatrix} \bar{\mu}_1^{(1)} x_0^{(1)} & \bar{\mu}_1^{(1)} x_{-s}^{(1)} & \cdots & \bar{\mu}_M^{(1)} x_{-ns}^{(1)} & \bar{\mu}_M^{(1)} \\ \bar{\mu}_1^{(2)} x_0^{(2)} & \bar{\mu}_1^{(2)} x_{-s}^{(2)} & \cdots & \bar{\mu}_M^{(2)} x_{-ns}^{(2)} & \bar{\mu}_M^{(2)} \\ & & \vdots & & \\ \bar{\mu}_1^{(P)} x_0^{(P)} & \bar{\mu}_1^{(P)} x_{-s}^{(P)} & \cdots & \bar{\mu}_M^{(P)} x_{-ns}^{(P)} & \bar{\mu}_M^{(P)} \end{bmatrix}; \tag{2.11}$$

$\boldsymbol{\theta}$ is the consequent parameter set whose elements are to be updated

$$\boldsymbol{\theta} = \begin{bmatrix} c_0^1 & c_1^1 & c_2^1 & \cdots & c_n^M & c_{n+1}^M \end{bmatrix}^T; \tag{2.12}$$

and $\boldsymbol{d}$ represents the vector of the desired system states

$$\mathbf{d} = \begin{bmatrix} d^{(1)} & d^{(2)} & \cdots & d^{(P)} \end{bmatrix}^T. \tag{2.13}$$

Because the row vectors in $\boldsymbol{R}$ and the associated elements in $\boldsymbol{d}$ are obtained sequentially, $\boldsymbol{\theta}$ in (2.10) can be computed recursively. For the objective function with respect to the adjusTable parameters $\boldsymbol{\theta}_t$ at the current time instant t,

$$E_2(\boldsymbol{\theta}_t) = \frac{1}{2}\sum_{t=0}^{P-1} \lambda \left[x_{+s}(\boldsymbol{\theta}_t) - d^{(t)} \right]^2 \tag{2.14}$$

where $x_{+s}(\boldsymbol{\theta}_t)$ is determined by (2.8); $\mathbf{R}_t$ in (2.11) is the resulting matrix from the corresponding fuzzy inference operation at time t. The LSE is computed by

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \mathbf{S}_{t+1}\mathbf{R}_t(d_t - \mathbf{R}_t^T\boldsymbol{\theta}_t) \tag{2.15}$$

$$\mathbf{S}_{t+1} = \frac{1}{\lambda}\left[\mathbf{S}_t - \mathbf{S}_t\mathbf{R}_t\left[\lambda I + \mathbf{R}_t^T\mathbf{S}_t\mathbf{R}_t \right]^{-1}\mathbf{R}_t^T\mathbf{S}_t \right]. \tag{2.16}$$

where $t = 0, 1, \ldots, P\text{-}1$. $\lambda \in [0.95 \;\; 1]$ is the weight factor. The optimal estimate of $\boldsymbol{\theta}$ is $\boldsymbol{\theta}_P$, whereas $\boldsymbol{\theta}_0 = \mathbf{0}$. The initial conditions for the covariance matrix $\mathbf{S}_t$ is $\mathbf{S}_0 = \alpha\mathbf{I}$, $\alpha \in [10^2 \;\; 10^6]$, and $\boldsymbol{I}$ is an identity matrix.

2.4.3 Simulation Results

Four different configurations are used to compare the performance of the novel paradigm. The first structure simulated is the structure mentioned in (2.5) with the recurrent weights equaling zero incorporating sequential inputs. The second structure is similar to the first with the exception of equal-step inputs as opposed to sequential step inputs. Structures 3 and 4 are similar to 1 and 2, respectively, with the exception of the recurrent links.

The suggested hybrid adaptive training method adopts two algorithms: the GD and the LSE.

In each training epoch, the fuzzy system parameters in the recurrent context are adaptively optimized by applying GD in the back pass, whereas the linear consequent parameters are updated by using the weighted LSE in the forward pass.

The GD algorithm trains the recurrent network while the network continues to perform its signal processing function, rather than at the end of the presented sequences. In initialization, the synaptic weights of the algorithm are set to small values (0.025 in this case) from a uniform distribution, while both the state vector and its partial derivative (with respect to the weight vector) are set to zero.

The comparison is performed by a series of simulation tests based on data sets from the Mackey-Glass equation [6]. The paradigms are tested with 4 inputs, incorporating time steps s = [3, 6, 9 and 12]. For each time step, the systems are trained over 100 epochs and 800 data points from the Mackey-Glass equation. Each test is repeated over 10 times to minimize random effects related to initial values of weights. After the training period, 400 data pairs are used for prediction verification. Table 2.3 presents the results.

Table 2.3. Mean square errors for the recurrent and neuro-fuzzy predictors.

		MSE - Steps			
	Inputs	3	6	9	12
NF	sequential	2.45E-05	3.33E-04	1.91E-04	1.55E-04
NF	equal	5.31E-07	2.22E-06	8.02E-06	7.13E-05
RNF	sequential	4.11E-05	9.20E-05	9.88E-05	1.06E-04
RNF	equal	1.85E-08	3.89E-07	3.50E-07	7.72E-06

In Figure 2.8 the original MFs and trained MFs are given for the recurrent NF paradigm. Figure 2.9 illustrates the results for 12 step-ahead prediction on the RNF paradigm. As can be seen, both illustratively and numerically, the weighted recurrent NF paradigm incorporating equal-step inputs outperforms all other suggested structures in this chapter. Applications of this novel paradigm can be found in [25].

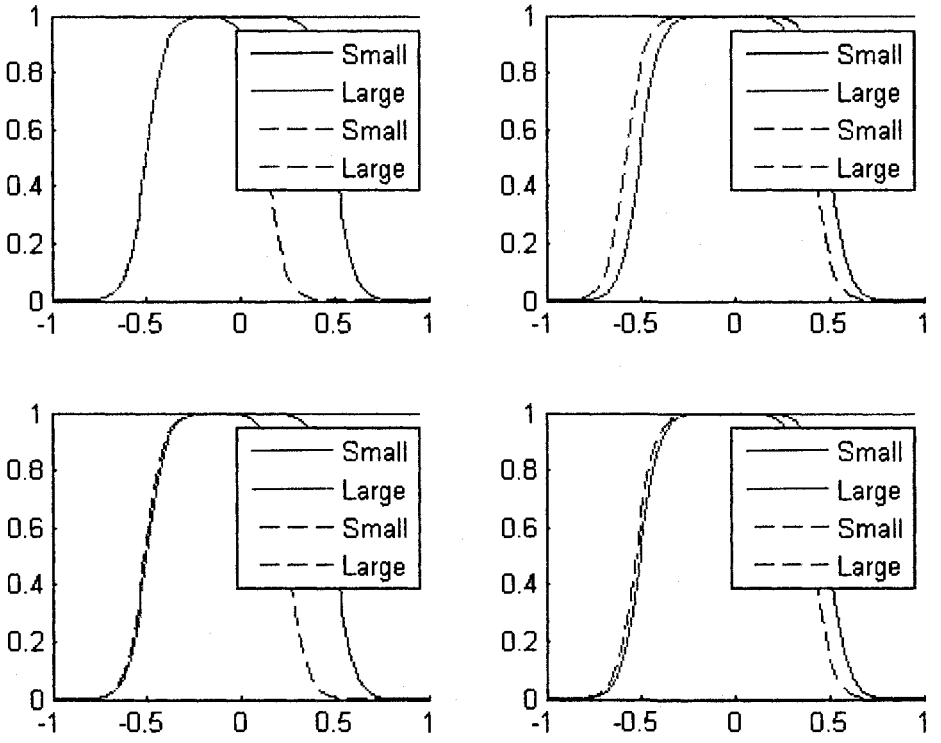

Figure 2.8. Membership Functions: before training (solid) and after training (dotted); (a) x_1 (top left); (b) x_2 (top right); (c) x_3 (bottom left) and (d) x_4 (bottom right).

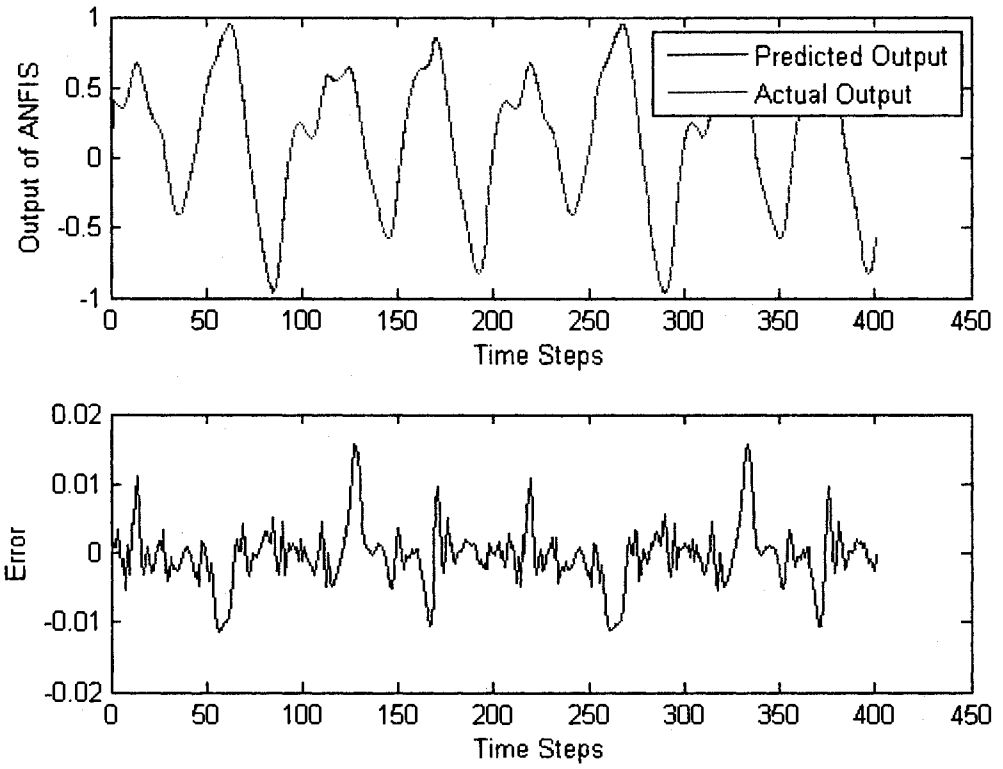

Figure 2.9. 12 step prediction via RNF by means of equal input steps.

2.5 Summary of the Results

Within this chapter, knowledge-based data-driven multi-step-ahead forecasting schemes are evaluated in terms of performance and efficiency. Of the feedforward NNs, recurrent NNs, and recurrent NF systems, analysis results have shown that predictors based on step inputs perform better than those based on sequential inputs, as long as the same number inputs are employed. Moreover, the recurrent NF predictor performs better than those based on feedforward and recurrent NNs, in multi-step-ahead forecasting operations. A summary of the results can be found in Figures 2.10 and 2.11.

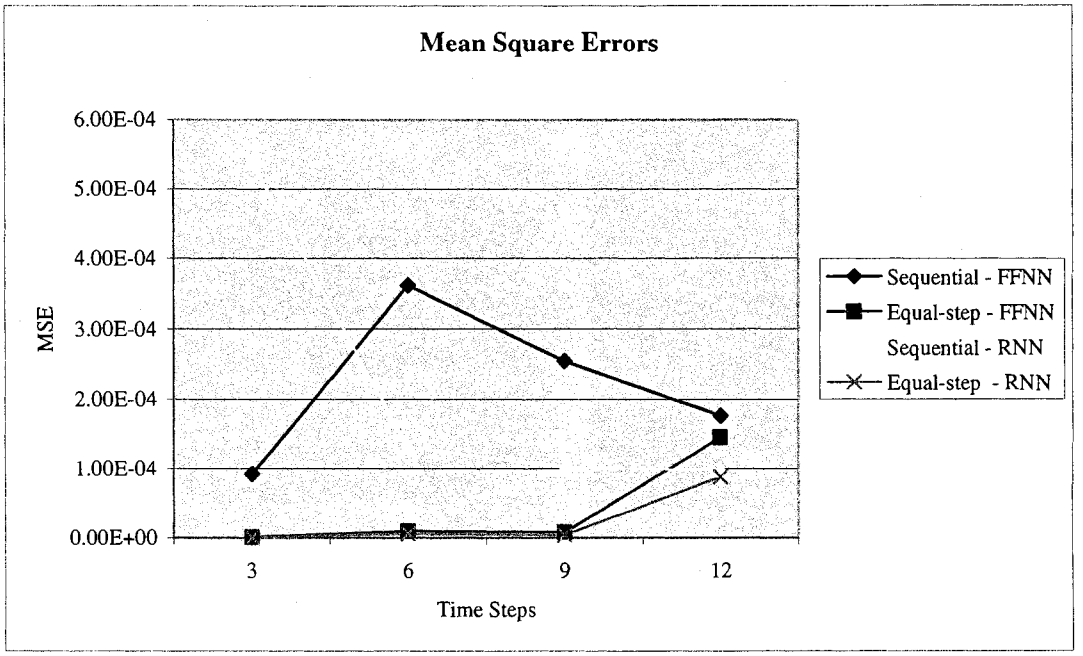

Figure 2.10. Mean square errors of the neural networks.

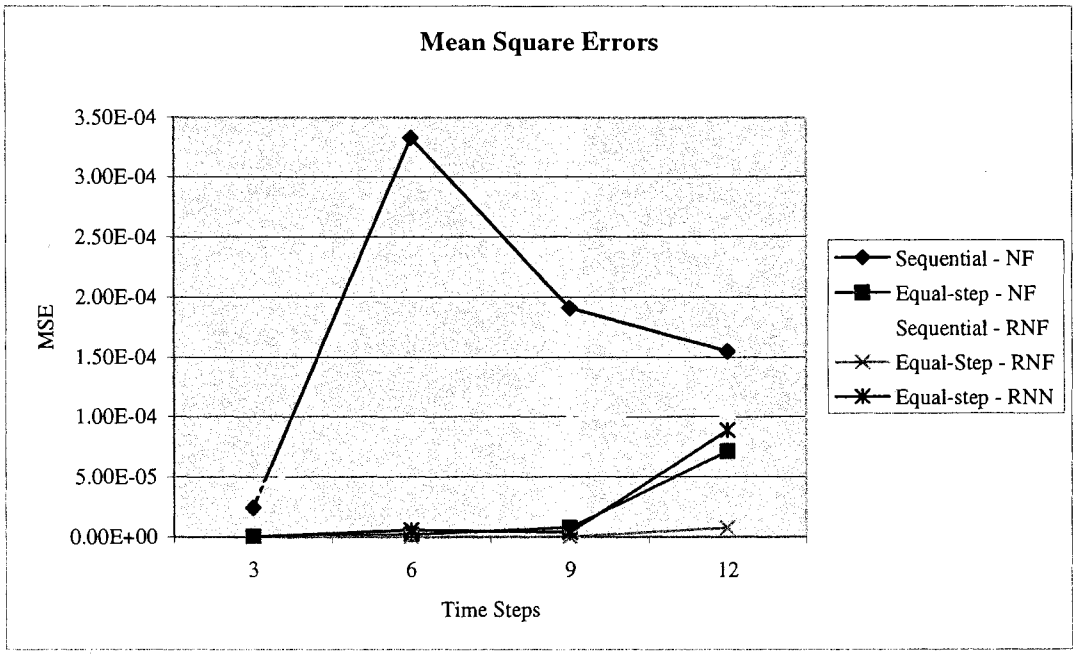

Figure 2.11. Mean square errors of the neural networks.

A hybrid training algorithm was adopted to improve the robustness and reliability of the recurrent NF predictor, by which the fuzzy parameters in the recurrent context layer are trained by using the GD algorithm, whereas the fuzzy consequent parameters are updated by applying a weighted LSE. The paradigm integrates the benefits of both GD and weighted LSE and posses the randomness that is helpful to escape from local minima.

Although the recurrent NF paradigm outperformed all the aforementioned structures, there exists one drawback; training time. Average training times for the feed-forward NN, recurrent NN, NF and recurrent NF paradigms are 12s, 22s, 12s and 17s, respectively. Although the times are relatively insignificant due to the speeds of current processors, it is worth noting, however, care must be taken if applying the novel design to on-line applications.

Despite the positive results in this chapter, problems exist when it comes to creating a NF structure. The current training is mainly for NF parameters, whereas its reasoning structure remains unchanged in training. Sometimes it is hard to determine, initially, an ideal fuzzy reasoning structure, for example, how many rules to create. Some priori knowledge is required in order to develop a more efficient and accurate predictor (or classifier).

In the next chapter, evolving NF paradigms are explored. Evolving paradigms are capable of generating their own rule-base resulting in a more accurate and robust structure.

Chapter 3

A NEW EVOLVING SCHEME WITH MAPPING CONSISTENCY AND CLUSTER COMPATABILITY

In Section 3.1, the literature survey in Chapter 1 is extended to introduce a brief introduction to some of the evolving NF structures in literature. In Section 3.2, a novel evolving clustering technique is introduced. The novel technique is compared to the potential-based eTS, or eTS[‡], method developed by Angelov *et al.* [8]. The two techniques are evaluated in real-time. In Section 3.3, a hybrid training technique is proposed and the two aforementioned structures are trained off-line. The structures are first established, and then the parameters are trained by the mentioned hybrid-technique.

3.1 Introduction to Evolving Structures

Evolving fuzzy systems originated from a self-organizing fuzzy inference system (SOFIS) suggested by Tanaka *et al.* [30]. They created a self-organizing algorithm based on the Takagi-Sugeno-Kang fuzzy model. The authors simplify a procedure for finding the optimal structure of fuzzy partition. The SOFIS consisted of four stages which effectively realized structure identification and parameter identification.

[‡] Potential-based eTS or eTS will be used interchangeably through this thesis.

The first three stages incorporate a simplified unbiased criterion that measures the sensitivity of parameters to the training set. In the last stage, the delta rules are used for adaptively modifying the related parameters of the system.

Juang *et al.* in paper [31], expanded on the idea of SOFIS by developing a NF system that exhibits some self-adaptive properties. They developed a self-constructing NF inference network (SONFIN) that is capable of on-line learning. The SONFIN is a modified Takagi-Sugeno-Kang type fuzzy rule-based model possessing NNs learning ability. The NNs learning ability allowed the authors to use RLSE for the consequent parameters and backpropagation for the premise parameters. In this way, they were able to adopt a paradigm for real-time applications. In this structure, the rule-base would grow in accordance to the partitioning of the input space, however, not at the rate of exponential growth.

Up to now, several clustering methods exist [32, 33, 34] and have been explored for partitioning the input and output spaces. An exhaustive review of NF rule generation can be found in [35].

3.2 The Evolving Fuzzy System (EFS) Structure

3.2.1 The EFS Clustering Procedure

The majority of clustering algorithms available today cluster the input/output data separately [32, 33, 34]. Clustering the data separately generates specific problems. The generated clusters, for example, may not be consistent if some training data are noise affected or outliers.

In this subsection, a novel clustering technique is proposed. In this scheme, both input and output patterns are clustered simultaneously, with the added constraint of *mapping consistency* and *cluster compatibility*. In this way, the noise affected data (or outliers) can be excluded resulting in more meaningful clusters (rules).

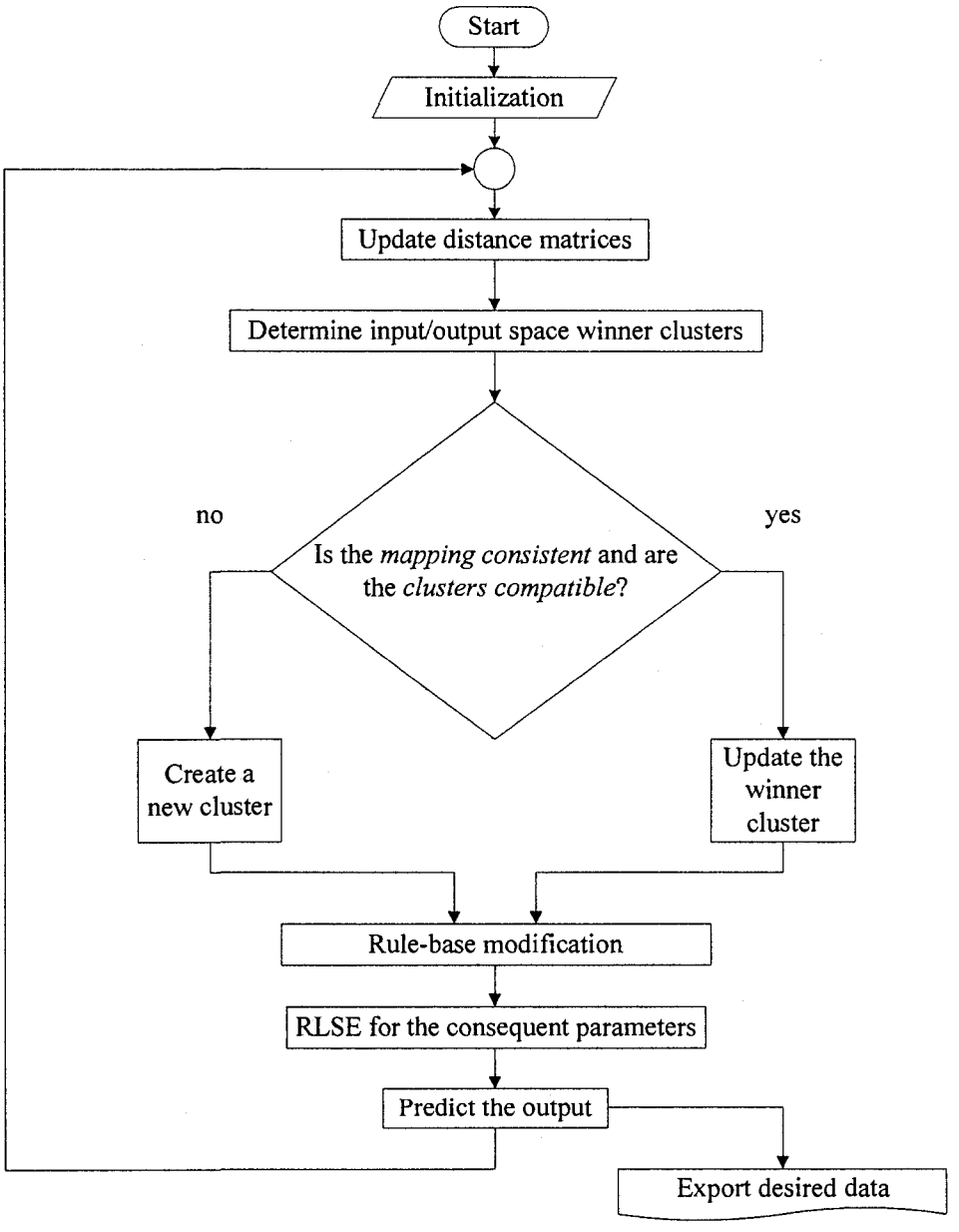

Figure 3.1. Flowchart for the EFS clustering algorithm.

The evolving clustering technique is data-driven, non-iterative, and a one-pass algorithm, based on the subtractive and mountain clustering techniques [3]. This technique extracts fuzzy rules gradually from the incoming data sets and the rule-base is updated based on specific criterion. A flowchart for the process is illustrated in Figure 3.1. The algorithm is as follows.

Step 1: *Initialization.*

Initially, the scheme starts with an empty rule-base. The first input is used to create the first rule. The data defines the cluster center and the initial widths of the cluster. That is:

$$\boldsymbol{m}_k^I \leftarrow \boldsymbol{x}_k,\ \boldsymbol{\sigma}_k^I \leftarrow \mathbf{0.1};\ \ \boldsymbol{m}_k^O \leftarrow \boldsymbol{y}_k,\ \ \boldsymbol{\sigma}_k^O \leftarrow \mathbf{0.1},$$

where $\boldsymbol{m}_k^I$, $\boldsymbol{m}_k^O$, $\boldsymbol{\sigma}_k^I$ and $\boldsymbol{\sigma}_k^O$ are the cluster centers and spreads in the input and output spaces, respectively, $k \in [1, K]$; The number of the clusters: $K \leftarrow 1$; The volume (i.e., the number of samples) in this cluster: $N_k \leftarrow 1$.

Step 2: *Distance matrix determination.*

$\boldsymbol{D}_I$ and $\boldsymbol{D}_O$, $\boldsymbol{D}_I \in \mathrm{R}^{K\times K}$ and $\boldsymbol{D}_O \in \mathrm{R}^{K\times K}$, are distance matrices in input and output spaces, respectively. The distance matrices represent the distance measurement between the centers of the corresponding clusters and are defined as

$$[\boldsymbol{D}_I]_{pg} = \left\|\boldsymbol{m}_p^I - \boldsymbol{m}_q^I\right\|^2 \text{ and } [\boldsymbol{D}_O]_{pq} = \left\|\boldsymbol{m}_p^O - \boldsymbol{m}_q^O\right\|^2, \text{ where } p, q \in [1, K]. \qquad (3.1)$$

Step 3: *Winner cluster determination.*

Within this step, the winning clusters, that is the closest clusters for an incoming data set, $[\boldsymbol{x}_t, \boldsymbol{y}_t]$, $t = 1, 2, \cdots, P$, $\boldsymbol{x} \in \mathrm{R}^n$, $\boldsymbol{y} \in \mathrm{R}^M$, are calculated. The winner clusters are calculated as follows

for the input space: $W_I = \arg\min_{k=1}^{K} \left\| \boldsymbol{m}_k^I - \boldsymbol{x}_t \right\|$; (3.2)

for the output space: $W_O = \arg\min_{k=1}^{K} \left\| \boldsymbol{m}_k^O - \boldsymbol{y}_t \right\|$. (3.3)

Step 4: *Structure recognition.*

The algorithm has two constraints in checking whether the sample belongs to an existing cluster; (1) mapping consistence and (2) cluster compatibility. If $W_I = W_O$, then mapping consistence holds. The cluster compatibility is measured by checking if the new incoming data set is highly descriptive by an existing cluster, or with a description grade larger than $\mu_{\min}$ (1/3 in this case). The description grade includes the area of the cluster from the maximum value to 1/3 of the maximum. If both conditions hold, merge the new data set to the existing cluster:

$K \leftarrow K$; The volume of the winner cluster: $N_W \leftarrow N_W + 1$.

The winner cluster parameters are modified to accurately reflect the change, whereas the other cluster information remains unchanged.

For the input space:

$$\left(\sigma_{W,t}^I\right)^2 \leftarrow \left(\sigma_{W,t-1}^I\right)^2 + \frac{1}{N_W}\left[\left(\boldsymbol{x}_t - \boldsymbol{m}_{W,t-1}^I\right)^2 - \left(\sigma_{W,t-1}^I\right)^2\right], \tag{3.4}$$

$$\boldsymbol{m}_{W,t}^I \leftarrow \boldsymbol{m}_{W,t-1}^I + \frac{\boldsymbol{x}_t - \boldsymbol{m}_{W,t-1}^I}{N_W}. \tag{3.5}$$

For the output space:

$$\left(\sigma_{W,t}^O\right)^2 \leftarrow \left(\sigma_{W,t-1}^O\right)^2 + \frac{1}{N_W}\left[\left(\boldsymbol{y}_t - \boldsymbol{m}_{W,t-1}^O\right)^2 - \left(\sigma_{W,t-1}^O\right)^2\right], \tag{3.6}$$

$$\boldsymbol{m}_{W,t}^O \leftarrow \boldsymbol{m}_{W,t-1}^O + \frac{\boldsymbol{y}_t - \boldsymbol{m}_{W,t-1}^O}{N_W}. \tag{3.7}$$

After the computation of equations (3.4) to (3.7), update the distance matrices $\boldsymbol{D}_I$ and $\boldsymbol{D}_O$.

Step 5: *If the criteria in **step 4** does not hold, create a new cluster.*

If the constraint criteria, mapping consistence and cluster compatibility, do not hold simultaneously, create a new cluster. That is

$$K \leftarrow K+1,\ N_K \leftarrow 1,$$

$$\boldsymbol{m}_K^I \leftarrow \boldsymbol{x}_t,\ \boldsymbol{\sigma}_K^I \leftarrow \mathbf{0.1} \text{ and } \boldsymbol{m}_K^O \leftarrow \boldsymbol{y}_k,\ \boldsymbol{\sigma}_K^O \leftarrow \mathbf{0.1}.$$

Upon creating a new cluster, update the distance matrices $\boldsymbol{D}_I$ and $\boldsymbol{D}_O$.

These two constraint criteria are applied to prevent contradictory rules. For example, two closest clusters may not be merged to one cluster if they belong to different classes (out of consistence).

Step 6: *Rule-base modification.*

It is common for some clusters to form, which may be compatible with existing clusters. Two clusters, α and β are compatible, if both $\boldsymbol{D}_{I,\alpha\beta}$ and $\boldsymbol{D}_{O,\alpha\beta}$ are minimum $(\alpha \neq \beta)$, in addition, to the clusters having a descriptive grade larger than $\mu_{\min}$. If this is the case, merge them to a new cluster, that is

$$\alpha \leftarrow \alpha+\beta,\ K \leftarrow K-1.$$

$$\left(\boldsymbol{\sigma}_\alpha^I\right)^2 \leftarrow \frac{N_\alpha\left(\boldsymbol{\sigma}_\alpha^I\right)^2 + N_\beta\left(\boldsymbol{\sigma}_\beta^I\right)^2}{N_\alpha+N_\beta} + \left(\frac{N_\alpha \boldsymbol{m}_\alpha^I - N_\beta \boldsymbol{m}_\beta^I}{N_\alpha+N_\beta}\right)^2, \tag{3.8}$$

$$\boldsymbol{m}_\alpha^I \leftarrow \frac{N_\alpha \boldsymbol{m}_\alpha^I + N_\beta \boldsymbol{m}_\beta^I}{N_\alpha+N_\beta}. \tag{3.9}$$

$$\left(\sigma_\alpha^O\right)^2 \leftarrow \frac{N_\alpha\left(\sigma_\alpha^O\right)^2 + N_\beta\left(\sigma_\beta^O\right)^2}{N_\alpha + N_\beta} + \left(\frac{N_\alpha \boldsymbol{m}_\alpha^O - N_\beta \boldsymbol{m}_\beta^O}{N_\alpha + N_\beta}\right)^2, \tag{3.10}$$

$$\boldsymbol{m}_\alpha^O \leftarrow \frac{N_\alpha \boldsymbol{m}_\alpha^O + N_\beta \boldsymbol{m}_\beta^O}{N_\alpha + N_\beta}, \tag{3.11}$$

$$N_\alpha \leftarrow N_\alpha + N_\beta. \tag{3.12}$$

Upon computation, update the distance matrices $\boldsymbol{D}_I$ and $\boldsymbol{D}_O$.

Step 7: *Training of the consequent parameters.*

The consequent parameters are trained via the RLSE as discussed in subsection 2.4.2.

Step 8: *Output prediction.*

The output for the next time step is predicted by

$$\hat{y}_{t+1} = \mathbf{R}_t^{\mathrm{T}}\boldsymbol{\theta}_t. \tag{3.13}$$

The algorithm continues from **Step 2** by reading the next data sample at the next time step.

3.2.2 Experimental Results

The new algorithm is compared to the potential-based eTS [8] algorithm which outperforms the Dynamic Evolving Neuro-Fuzzy Inference System (DENFIS) [36], a Resource Allocation Network (RAL) [37], an Evolving Self-Organizing Mapping system (ESOM) [38] and an Evolving Fuzzy Neural Network (EFuNN) [39]. Performance is based on the number of rules generated, the mean-square error (MSE) of the verification data and the time it takes to iterate the input data. The two algorithms, the new approach and the eTS, are compared based on data sets from the Mackey-Glass [6] time series.

The inputs are sequential and the prediction is 10 steps-ahead. Sixteen hundred data sets are used for training and 1000 are used for verification.

Figures 3.2 and 3.3 illustrate the rules (clusters) in 2 dimensions with respect to the first two and last two outputs, respectively. The algorithm generated ten rules based on the number of data samples used for this experiment. The evolving rule-base is shown in Figure 3.4. Figure 3.5 depicts the MFs for all the input variables with respect to the number of rules.

Once the structure is established, 1000 data sets are entered into the system for verification. At this point, no parameters are trained. The result of the simulation is demonstrated in Figure 3.6.

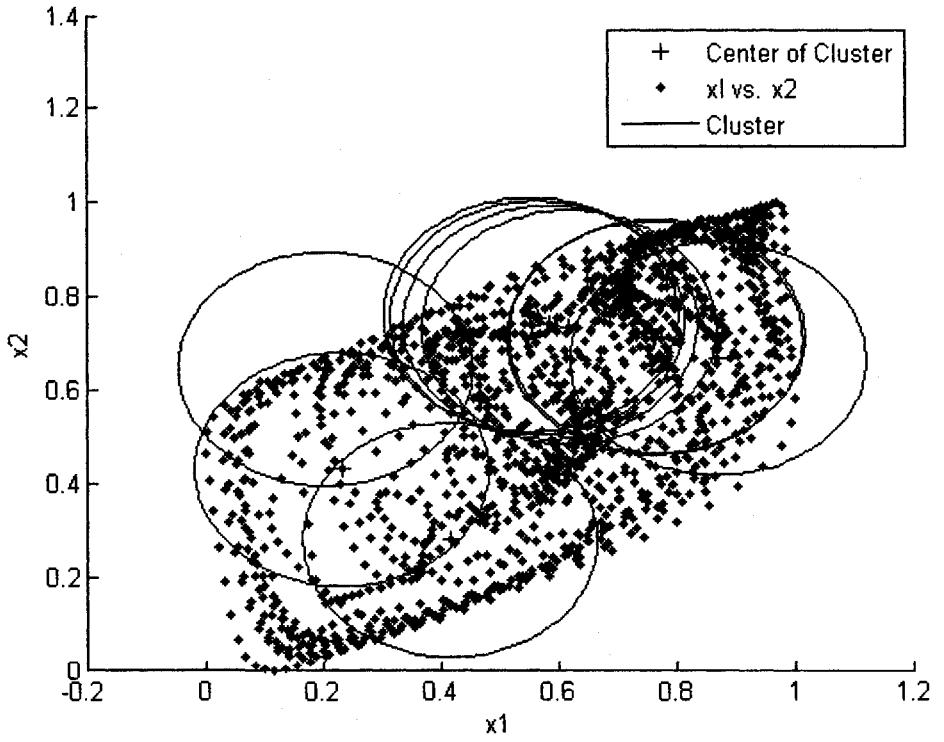

Figure 3.2. 2-D illustration (x_1 vs. x_2) of the rule-base for the eTS structure.

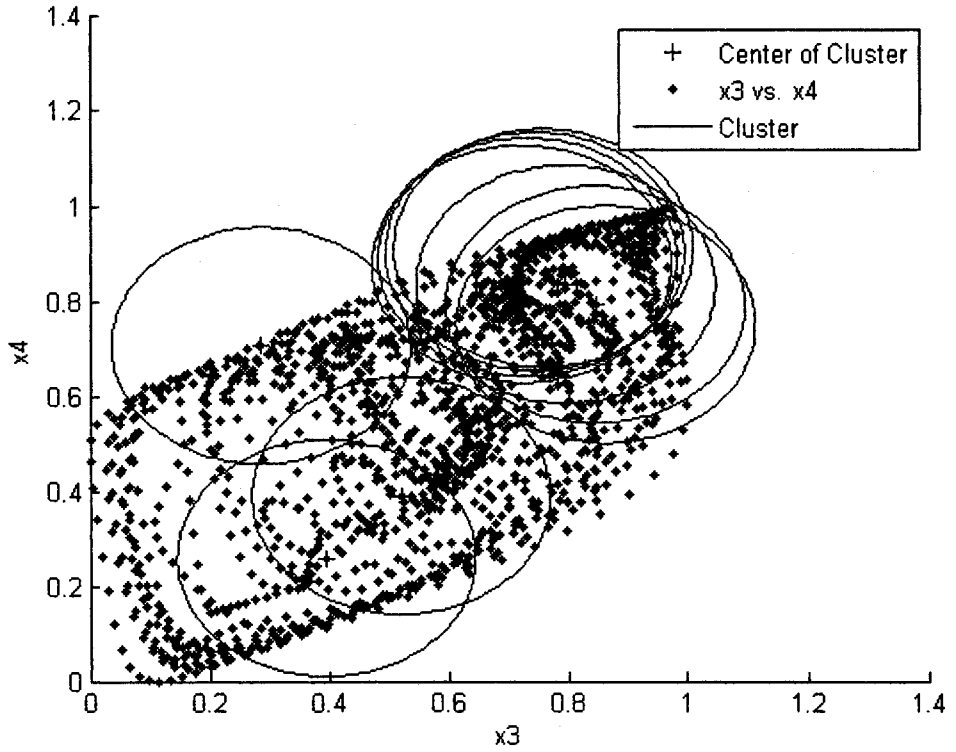

Figure 3.3. 2-D illustration (x_3 vs. x_4) of the rule-base for the eTS structure.

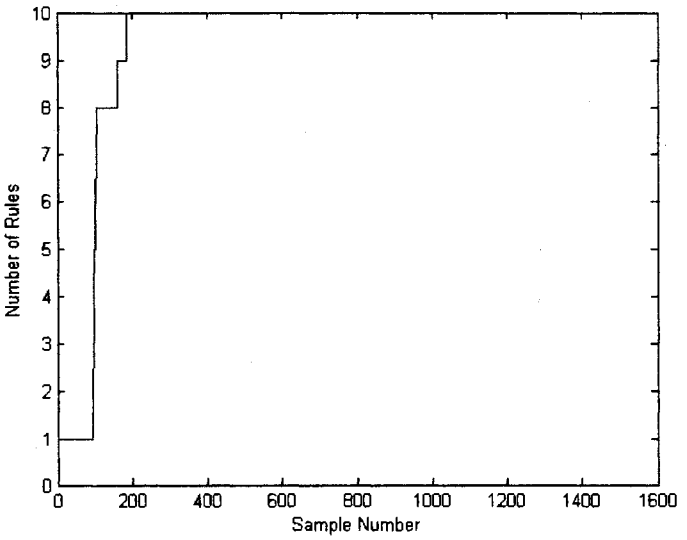

Figure 3.4. Evolving rule-base for the eTS structure.

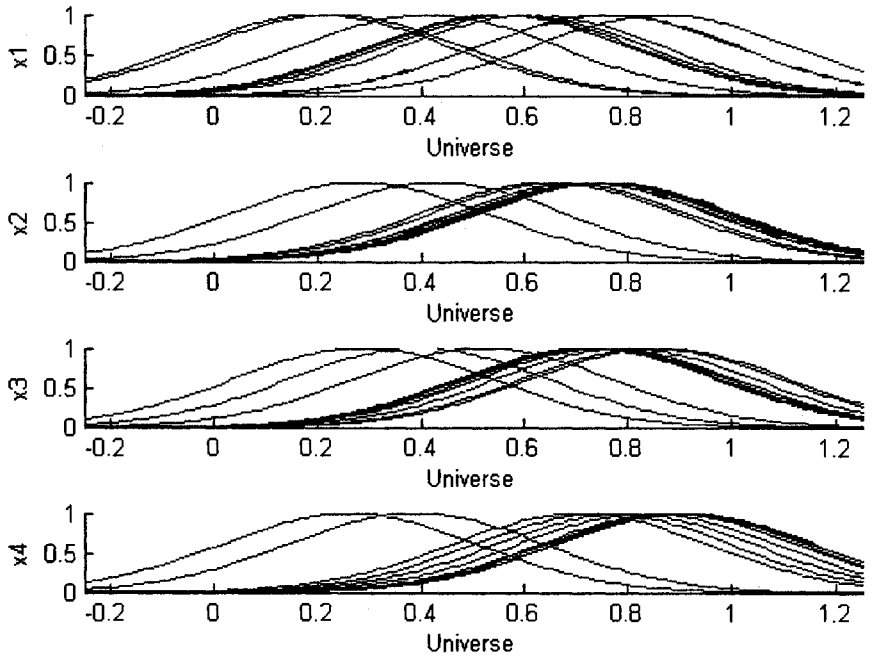

Figure 3.5. MFs for the eTS structure.

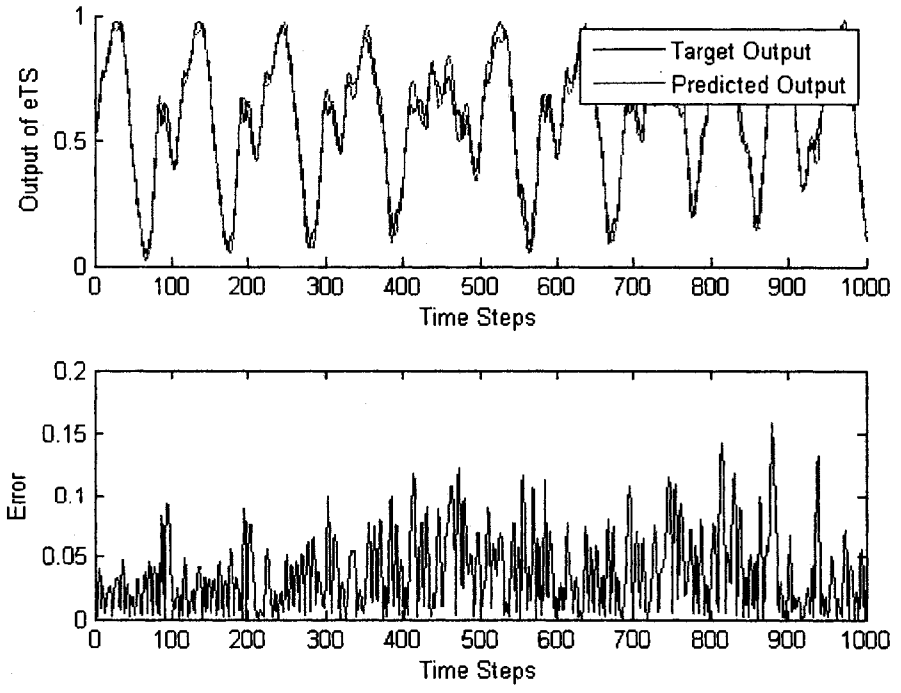

Figure 3.6. The target and predicted outputs of the eTS scheme.

The new EFS is simulated with the same number of data sets for online training and verification. Figures 3.7 and 3.8 illustrate the rules (clusters) in 2 dimensions with respect to the first two and last two outputs, respectively.

The clusters in the EFS are much different in size and location than the clusters produced by the eTS scheme. The difference in location and size are due to the recursive update equations (3.4) to (3.7).

The numbers of rules generated are significantly less than that of the eTS scheme. This is due to the fact that the mapping consistency criterion and the cluster compatibility criterion eliminate the outlying clusters.

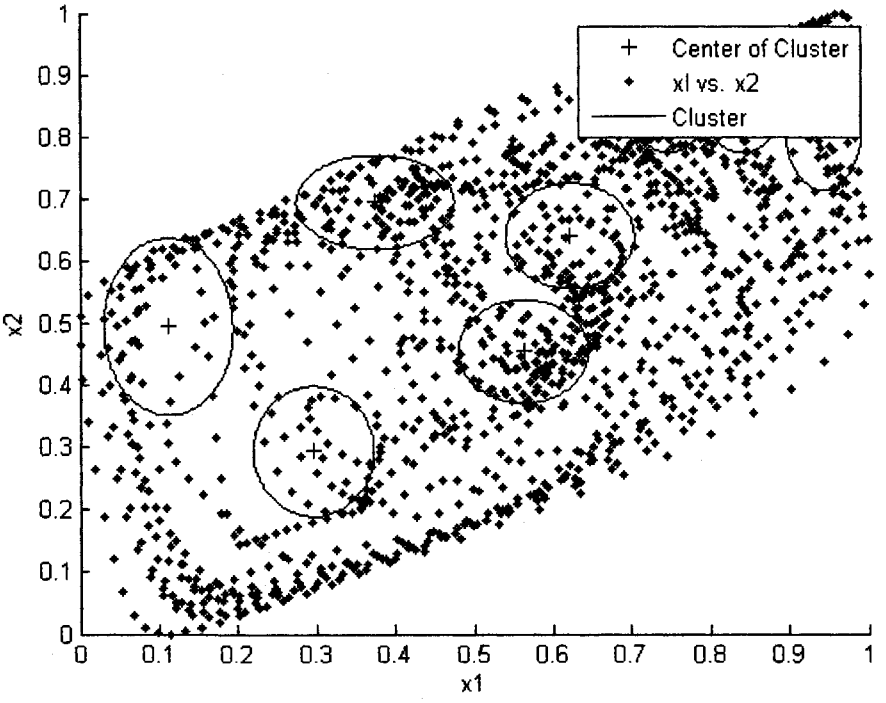

Figure 3.7. 2-D illustration (x_1 vs. x_2) of the rule-base for the EFS structure.

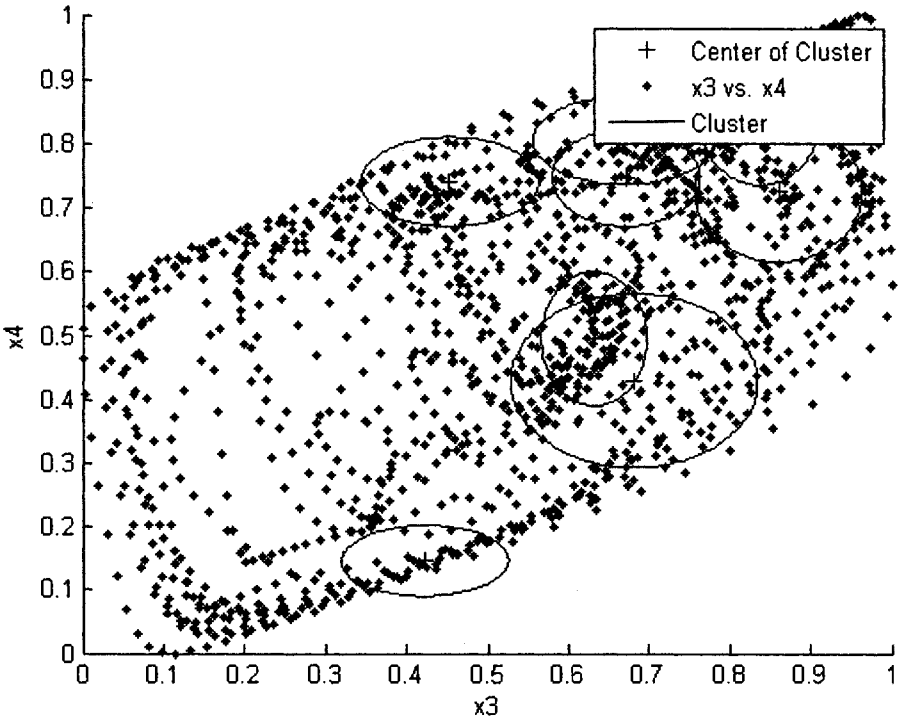

Figure 3.8. 2-D illustration (x_3 vs. x_4) of the rule-base for the EFS structure.

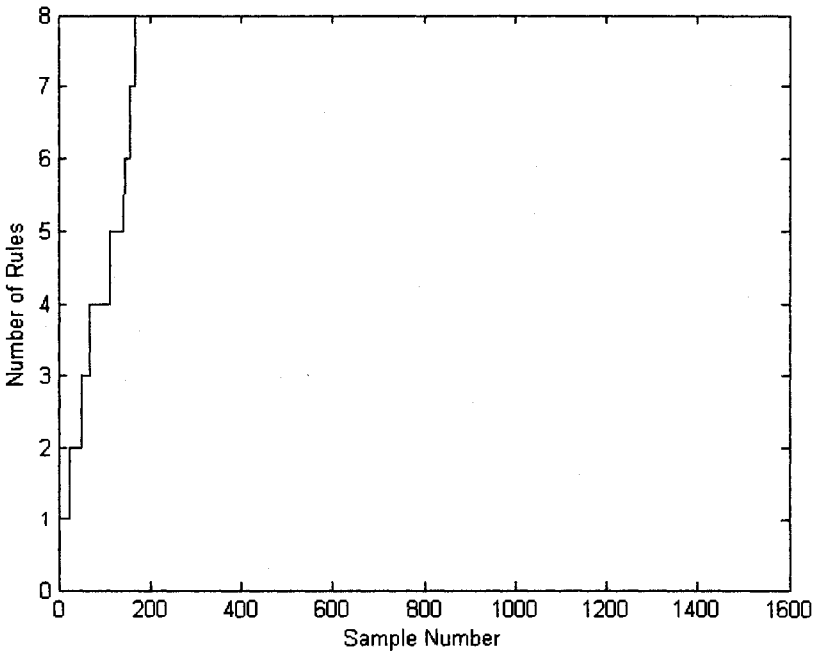

Figure 3.9. Evolving rule-base for the EFS structure.

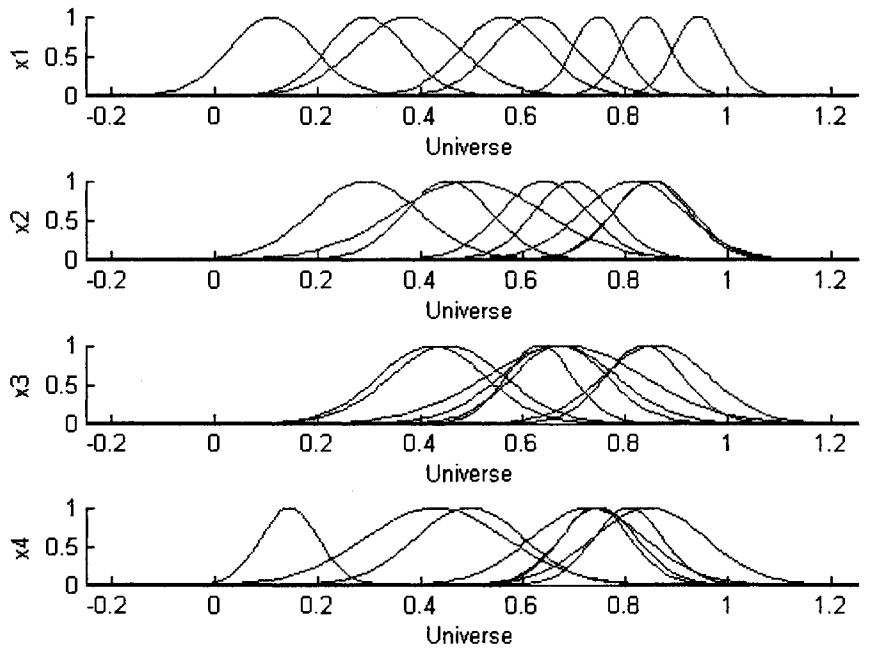

Figure 3.10. MFs for the eTS structure.

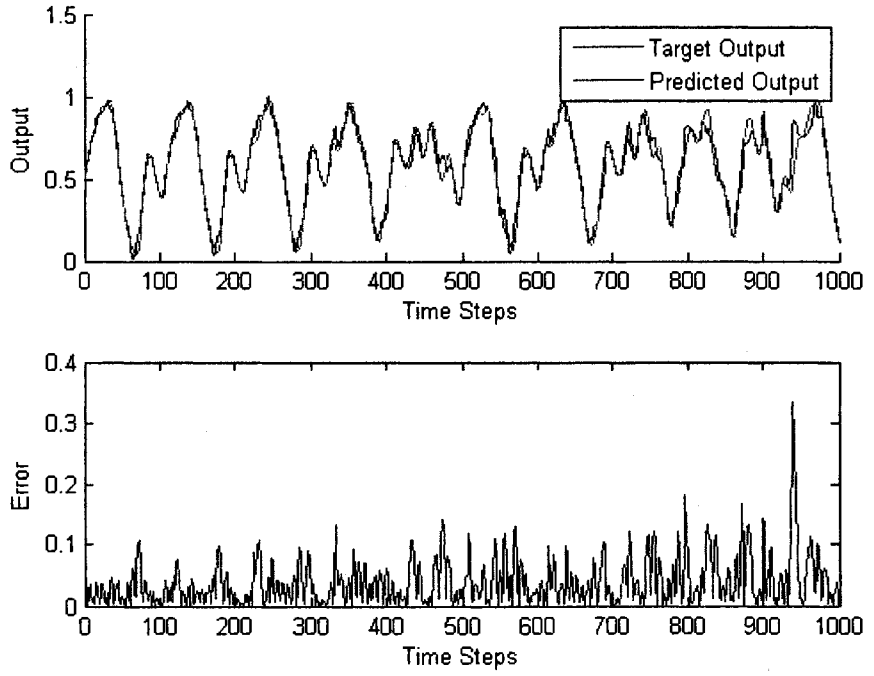

Figure 3.11. The target and predicted outputs of the EFS scheme.

The MFs for the EFS structure are better defined; they represent the input/output mapping with more accuracy. This is due to the fact that the MFs are iteratively adjusted for the new data sets and any required modification of the rules.

Figure 3.11 depicts the results for the EFS scheme. Although the EFS scheme generated less error than the eTS scheme, with of MSE 0.00299 compared to 0.00349, the iteration times for the complete data sets leans towards favoring the eTS structure (1.771 s to 1.800 s). However, it is worth noting, the difference in the iteration time is considered insignificant for most real-time applications [8].

3.3 Off-line Structure Identification

In this section, the EFS developed in section 3.1 is trained through a Recursive Levenberg-Marquardt (RLM)-RLSE hybrid training technique after the structure is identified on-line. Once the structure of the EFS is recognized, its premise and consequent parameters are trained via the RLM and RLSE training techniques, respectively. Training in this way will allow for optimization of the premise and consequent parameters. Once the parameters are optimized, the EFS structure will be simulated through the use of verification data for a performance evaluation.

The performance of the new EFS structure and training technique will be compared to the eTS structure incorporating the GD-RLSE and RLM-RLSE training techniques.

3.3.1 Introduction of the RLM-RLSE hybrid technique

For the sake of clarity, the following derivation of the RLM-RLSE will be based on a multi-input and single-output (MISO) system.

After the structure has been determined by the algorithm described in section 3.1, the ESF paradigm then performs the parameter identification to further tune the premise and consequent parameters of the existing structure.

The error function of a NF system represents the error of the predicted (classified) output amongst the desired value. The function incorporates the system's equations and, in this case, the error function with respect to adjustable parameters $\boldsymbol{\theta}_t$ at the current time state, t, is

$$E(\boldsymbol{\theta}_t) = \frac{1}{2}\sum_{t=1}^{P}[y_t(\boldsymbol{\theta}_t) - d_t]^2 = \frac{1}{2}\sum_{t=1}^{P} r_t^2(\boldsymbol{\theta}_t) = \boldsymbol{r}_t^T(\boldsymbol{\theta}_t)\boldsymbol{r}_t(\boldsymbol{\theta}_t) \tag{3.14}$$

where $y_t(\boldsymbol{\theta}_t)$ is the t^{th} output, $t = 1, 2, \cdots, P$; d_t is the desired output; and $\boldsymbol{r}_t$ is the error vector. If $\boldsymbol{r}_t$ is linear, (3.14) becomes a linear optimization problem whose global minimum can be found by a least-squares method. If the vector $\boldsymbol{r}_t$ is nonlinear in nature, as in this case, it is a non-linear least-squares optimization problem that usually arises in the phase of parameter learning due to the non-linear nature of the MFs [10].

By observing the EFS structure, the premise weights are non-linear in nature whereas the consequent weights are linear in nature. Ergo, the premise weights will be trained via the RLM algorithm in the backward pass and the consequent parameters will be trained via RLSE in the forward pass.

Taking the Taylor series, then, of (3.14) about the current parameter [10], and neglecting the higher order terms, (3.14) becomes

$$\boldsymbol{\theta}_{t+1} \approx \boldsymbol{\theta}_t + \lambda_t(\boldsymbol{J}_t^T\boldsymbol{J}_t + \eta_t\boldsymbol{I})^{-1}\boldsymbol{J}_t^T\boldsymbol{r}_t = \boldsymbol{\theta}_t + \lambda_t\boldsymbol{H}_t^{-1}\boldsymbol{J}_t^T\boldsymbol{r}_t = \boldsymbol{\theta}_t + (1-\alpha_t)\boldsymbol{H}_t^{-1}\boldsymbol{J}_t^T\boldsymbol{r}_t \tag{3.15}$$

where $\lambda_t = 1-\alpha_t$ is the learning rate, the Jacobian matrix $\boldsymbol{J}_t \in \mathrm{R}^{N\times Z}$, $\mathbf{J}_t = \dfrac{\partial^2 \mathbf{r}(\theta_t)}{\partial \theta_t}$ and Z is dimension (or the number of adjustable parameters) of $\boldsymbol{\theta}_t$. $\boldsymbol{H}_t \in \mathrm{R}^{Z\times Z}$ is the modified Hessian matrix $\boldsymbol{H}_t = \alpha_t \boldsymbol{H}_{t-1} - (1-\alpha_t)(\boldsymbol{J}_t^T \boldsymbol{J}_t + \eta_t \boldsymbol{I})$; $\boldsymbol{I} \in \mathrm{R}^{Z\times Z}$ is an identity matrix and η_t is the forgetting factor.

When the scalar η_t is zero, (3.15) becomes Gauss-Newton's method; when η_t is large, (3.15) becomes a gradient algorithm with a small step size [3].

The forgetting factor η_t can be optimized by line search or a trust region algorithm [40]. Both methods try to determine a region, in which the nonlinear problem is adequately represented by a quadratic model, so that convergence of the algorithm to a local minimum can be guaranteed.

One major problem does exist with the direct implementation of (3.15). The computational inverse of the Hessian matrix makes it impractical for online applications. To solve this problem, the matrix inversion lemma [41] will be applied to avoid the computation of the inverse Hessian matrix. The lemma states that

$$(\boldsymbol{A}+\boldsymbol{BCD})^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{B}(\boldsymbol{C}^{-1}+\boldsymbol{DA}^{-1}\boldsymbol{B})^{-1}\boldsymbol{DA}^{-1}, \tag{3.16}$$

where $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ and $\boldsymbol{D}$ are matrices such that $\boldsymbol{A}$ and $(\boldsymbol{C}^{-1}+\boldsymbol{DA}^{-1}\boldsymbol{B})$ are nonsingular matrices.

Based on (3.15), (3.16) can be rewritten as

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + (1-\alpha_t)\boldsymbol{H}_t^{-1}\boldsymbol{J}_t^T \boldsymbol{r}_t = \boldsymbol{\theta}_t + (1-\alpha_t)\times \tag{3.17}$$

$$\left\{(\alpha_t \boldsymbol{H}_{t-1})^{-1} - (\alpha_t \boldsymbol{H}_{t-1})^{-1}(1-\alpha_t)\boldsymbol{U}\left[\boldsymbol{V} + \boldsymbol{U}^T(\alpha_t \boldsymbol{H}_{t-1})^{-1}(1-\alpha_t)\boldsymbol{U}\right]^{-1}\boldsymbol{U}^T(\alpha_t \boldsymbol{H}_{t-1})^{-1}\right\}\boldsymbol{J}_t^T \boldsymbol{r}_t.$$

However, a modification of the term $\mathbf{J}_t^T\mathbf{J}_t + \eta_t\mathbf{I}$ in (3.15) has to be made so that the lemma can be used. One possible solution to this problem, as suggested in [41], is to add one diagonal element at a time of the $\eta_t\mathbf{I}$ matrix; as opposed to adding $\eta_t\mathbf{I}$ at each time step. Thus, the Hessian matrix can be expressed as

$$\boldsymbol{H}_t = \alpha_t \boldsymbol{H}_{t-1} - (1-\alpha_t)(\boldsymbol{J}_t^T\boldsymbol{J}_t + Z\eta_t\boldsymbol{\Lambda}) \tag{3.18}$$

where $\boldsymbol{\Lambda} \in \mathrm{R}^{Z\times Z}$ has only one nonzero element located at $t\ \{\mathrm{mod}(Z)+1\}$ diagonal position, or

$$\Lambda_{ii} = \begin{cases} 1 & if\ \ i = t\ \{\mathrm{mod}(Z)+1\} \\ 0 & otherwise \end{cases} \tag{3.19}$$

Alternatively, (3.18) can be rewritten as

$$\boldsymbol{H}_t = \alpha_t \boldsymbol{H}_{t-1} - (1-\alpha_t)[\boldsymbol{U}\boldsymbol{V}^{-1}\boldsymbol{U}^T] \tag{3.20}$$

where $\boldsymbol{U}$ is a $Z\times 2$ matrix with the first column corresponding to $\boldsymbol{J}_t$ and the second column consisting of a $Z\times 1$ vector with one element to 1 at the position of $\{t \bmod(Z)+1\}$

Hence,

$$\boldsymbol{U}^T(\boldsymbol{\theta}_t) = \begin{bmatrix} \boldsymbol{J}_t^T \\ 0\ \cdots\ 0\ 1\ 0\ \cdots\ 0 \end{bmatrix} \quad \text{and} \tag{3.21}$$

$$\boldsymbol{V}^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & Z\eta_t \end{bmatrix}. \tag{3.22}$$

Assume $\boldsymbol{A} = \alpha_t\boldsymbol{H}_{t-1}$, $\boldsymbol{B} = (1-\alpha_t)\boldsymbol{U}$, $\boldsymbol{C} = \boldsymbol{V}^{-1}$, and $\boldsymbol{D} = \boldsymbol{U}^T$. Then based on (3.17), the RLM algorithm can be represented by

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \boldsymbol{\Phi}_t \boldsymbol{J}_t \boldsymbol{r}_t \tag{3.23}$$

$$\boldsymbol{\Phi}_t = \frac{1}{\alpha_t}\left[\boldsymbol{\Phi}_{t-1} - \boldsymbol{\Phi}_{t-1}\mathbf{U}\left[\alpha_t \mathbf{V} + \mathbf{U}^T\boldsymbol{\Phi}_{t-1}\mathbf{U}\right]^{-1}\mathbf{U}^T\boldsymbol{\Phi}_{t-1}\right]. \tag{3.24}$$

The term $\alpha_t \boldsymbol{V} + \boldsymbol{U}^T\boldsymbol{\Phi}_{t-1}\boldsymbol{U}$ is a matrix with dimension 2×2; its inverse computation is simple, and can be implemented for real-time applications. $\alpha_t \in [0.95, 1]$ is the forgetting factor, $\alpha_t = 0.995$ in this case. $\boldsymbol{\theta}_0 = \mathbf{0}$; $\boldsymbol{\Phi}_t$ is a covariance matrix with initial condition $\boldsymbol{\Phi}_0 = \rho \boldsymbol{I}$, where $\rho \in [10^2, 10^5]$ is a positive quantity.

The above derived RLM algorithm will be used in the backward pass to train the nonlinear system parameters. In the forward pass, the consequent parameters will be trained using the RLSE [3].

3.3.2 Experimental Results

The performance of the new EFS structure and RLM-RLSE hybrid technique is evaluated based one of the better known benchmark data sets in research of classification; the iris data.

The Fisher-Anderson iris data consists of four input features, sepal length (*sl*), sepal width (*sw*), petal length (*pl*) and petal width (*pw*), on 150 specimens of an iris plant. Within this experiment, three species are involved, Iris Sestosa, Iris Versiolor and Iris Virginica where each specimen contains 50 instances.

To evaluate the performance of the new EFS structure, 75 instances have randomly been taken for structure identification (establishing the rule base), 75 instances have been randomly taken for parameter optimization (consequent and premise parameters) and the remaining 75 samples have been used for verification.

The data set was normalized to the range [0, 1]. To perform the classification, the output y of the EFS structure used the following classification rule:

$$Iris = \begin{cases} Sestosa, & \text{if } y < 0.33 \\ Versiolor, & \text{if } 0.33 \le y < 0.67 \\ Virginica, & \text{if } 0.67 \le y. \end{cases} \tag{3.25}$$

The new EFS structure and hybrid learning technique is compared to the potential-based eTS [8] algorithm. Through the literature, the eTS algorithm was proven to outperform the majority of classifiers and predictors [36, 37, 38 and 39]. Performance is based on the number of rules generated, the amount of training and testing errors and the ability of the system to properly classify data.

Figures 3.12 and 3.13 illustrate the results of the potential-based eTS clustering algorithm. Although the rules are represented in a multi-dimensional input space, for simplicity, the rules (clusters) are represented in 2-dimensions. The number of rules generated during the first random 75 instances of iris data is four. How the rules evolved based on the input data can be depicted in Figure 3.14.

The eTS structure did an excellent job in describing the rule-base. This can be seen in the lack of deviation of the number of rules after 28 instances. The initial (before parameter optimization) MFs corresponding to the individual inputs are depicted in Figure 3.15. Next, the aforementioned structure will be subjected to GD-RLSE and RLM-RLSE parameter optimization.

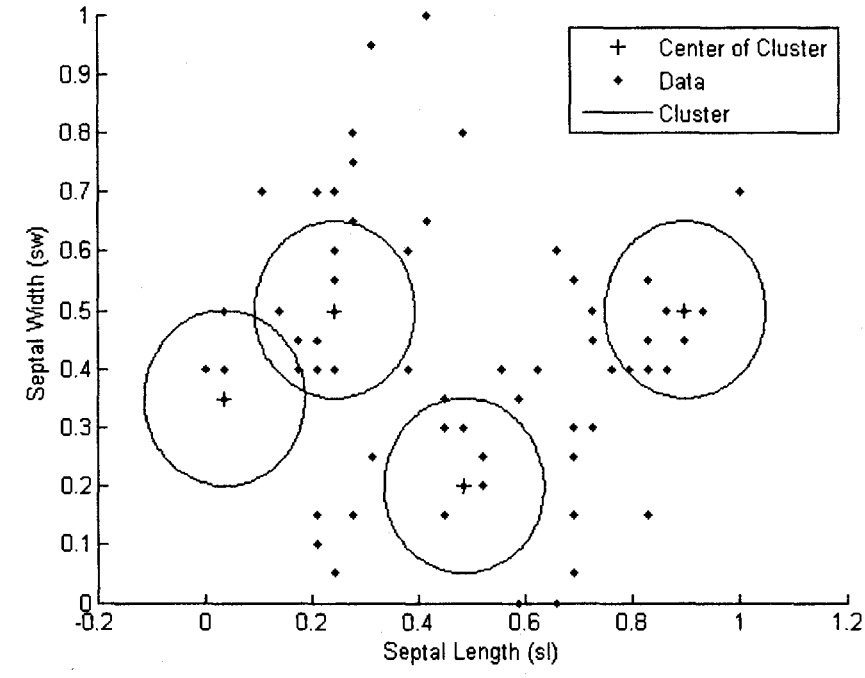

Figure 3.12. Sepal Length vs. Sepal Width for the eTS Scheme.

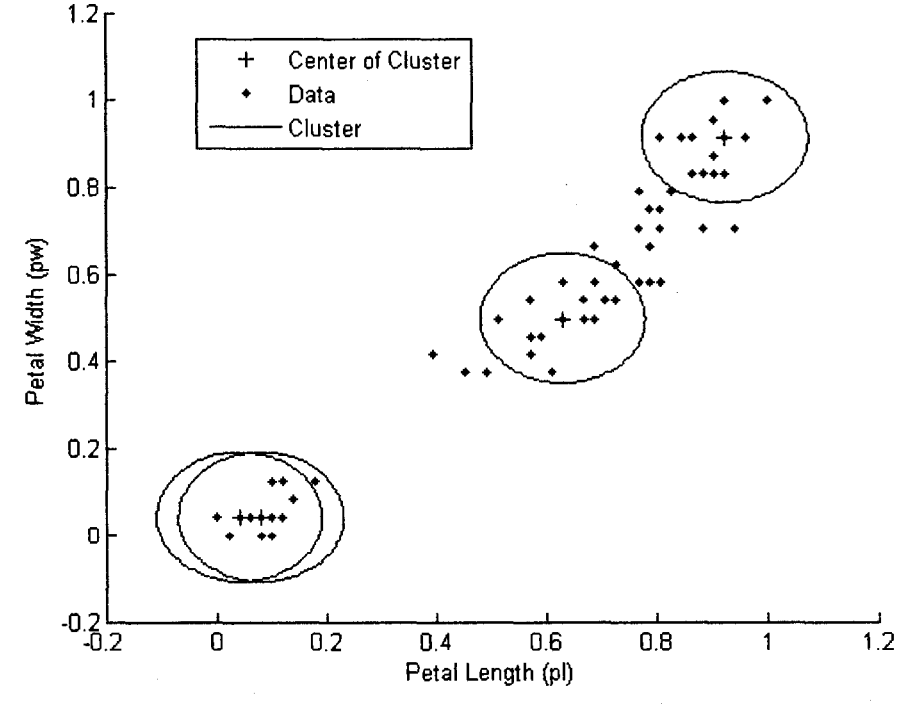

Figure 3.13. Petal Length vs. Petal Width for the eTS Scheme.

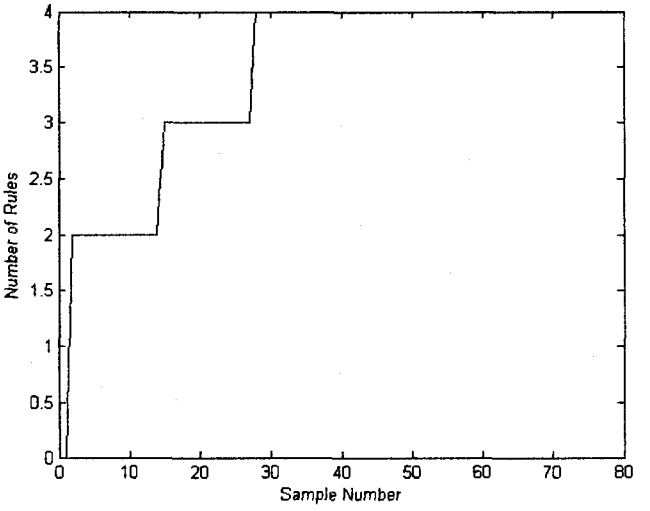

Figure 3.14. Evolving rule-base for the eTS Scheme.

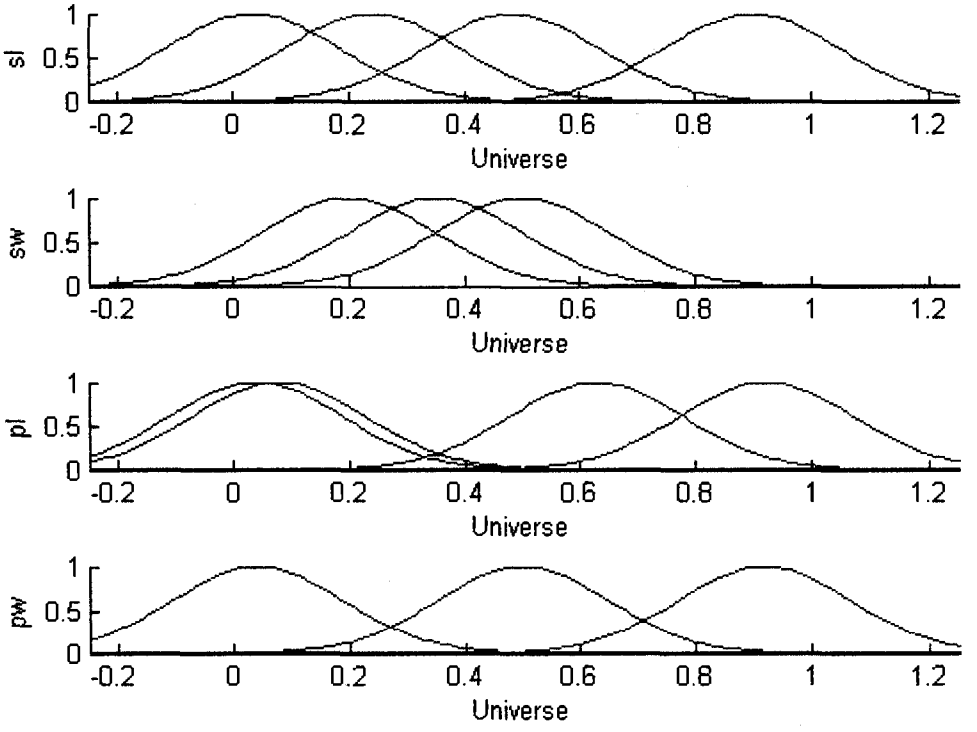

Figure 3.15. Initial MFs for the eTS Scheme.

Figure 3.16 shows the recognized structure, through GD-RLSE, and the corresponding MFs of each feature after the parameter learning phase and 75 random instances of iris data. One hundred epochs were used for parameter training. Within the training process, some of the MFs amalgamated. This is because some of the term sets were highly similar. The fuzzy rules of the iris classification can be extracted from the learned eTS structure from Figure 3.15 as follows.

1) IF *pw* is medium AND *pl* is small AND *sw* is medium AND *sl* is medium, THEN Iris is *Versicolor.*

2) IF *pw* is large AND *pl* is small AND *sw* is large AND *sl* is large, THEN Iris is *Virginica.*

3) IF *pw* is small AND *pl* is medium AND *sw* is small AND *sl* is small, THEN Iris is *Sestosa.*

Figure 3.17 illustrates the final output of the eTS structure incorporating the GD-RLSE hybrid training algorithm. The amount of testing errors accumulated to 16. The recognition rate, based on 75 instances for verification, is 78.67%.

Figure 3.18 shows the identified structure, through RLM-RLSE, and the corresponding MFs of each feature after the parameter learning phase and 75 random instances of iris data. One hundred epochs were used for parameter training. Within the training process, as in the case of GD-RLSE, some of the MFs amalgamated.

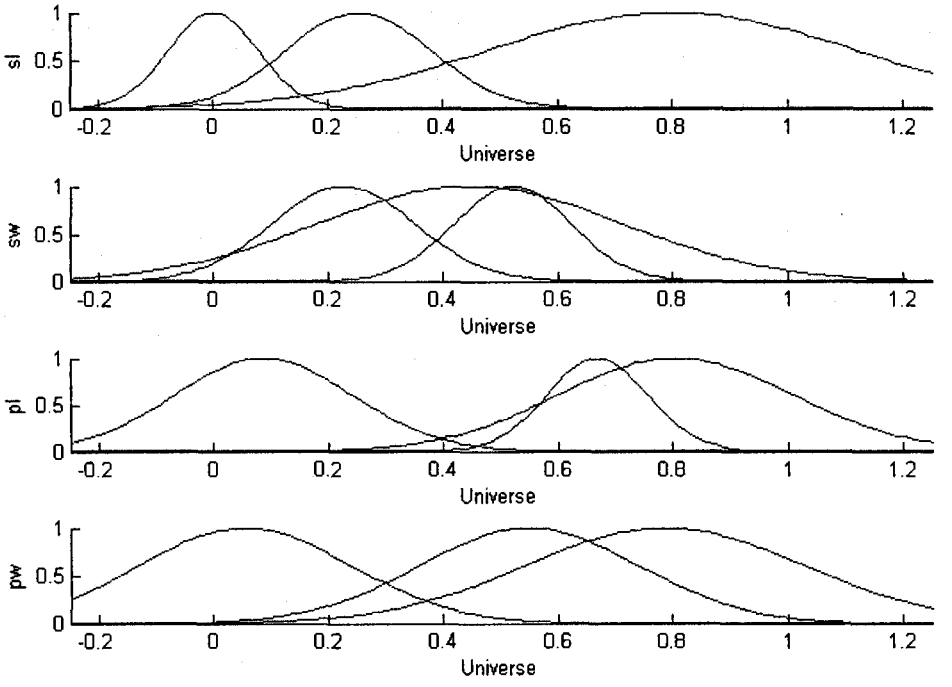

Figure 3.16. Final MFs for the eTS Scheme via GD-RSLE.

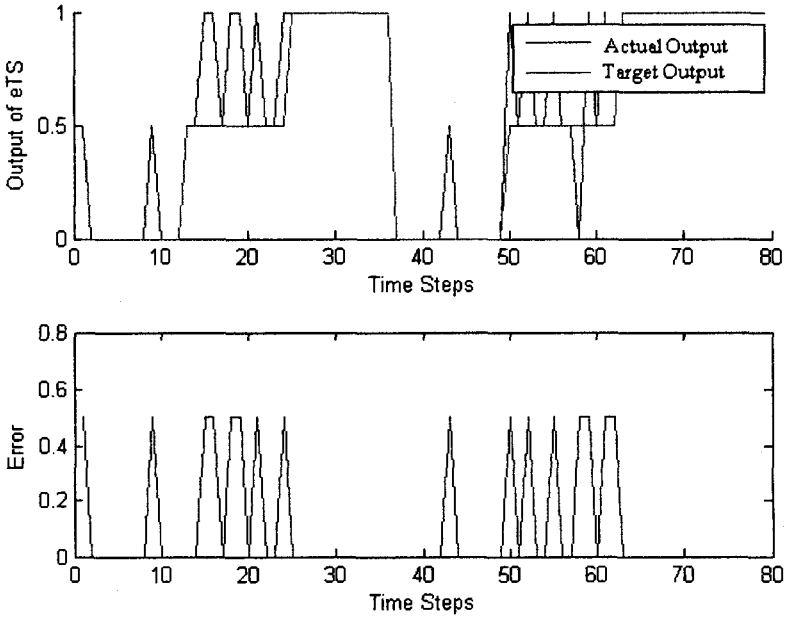

Figure 3.17. Final output for the eTS via GD-RSLE.

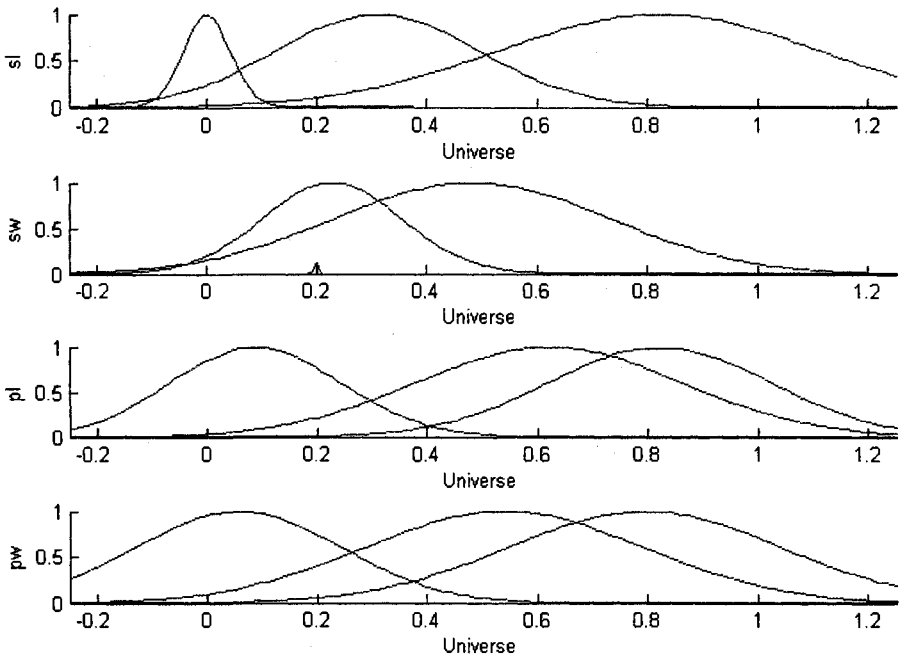

Figure 3.18. Final MFs for the eTS Scheme via RLM-RSLE.

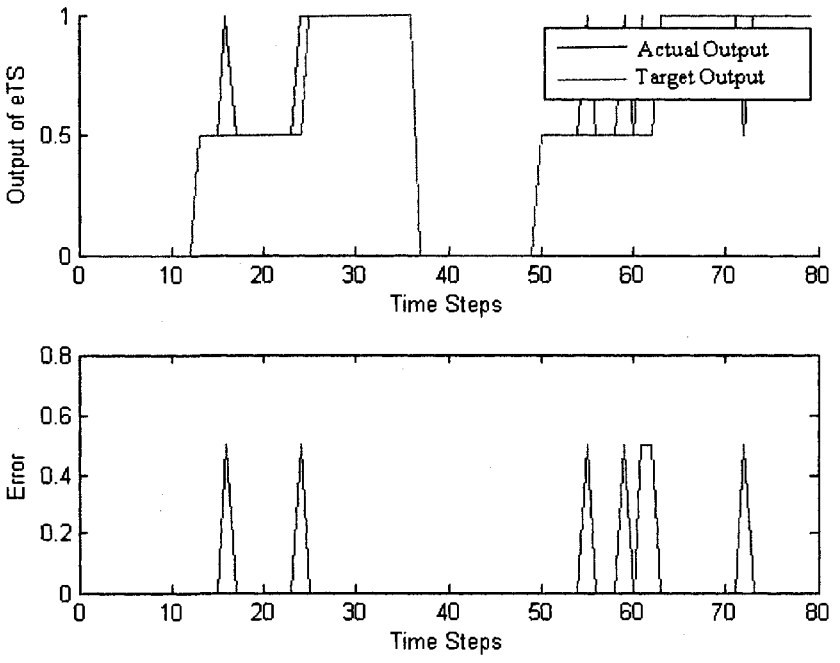

Figure 3.19. Final output for the eTS via RLM-RSLE.

The fuzzy rules of the iris classification can be extracted from the learned eTS structure from Figure 3.18 as follows.

1) IF *pw* is large AND *pl* is medium AND *sw* is medium AND *sl* is medium, THEN Iris is *Versiolor.*

2) IF *pw* is medium AND *pl* is large AND *sw* is large AND *sl* is large, THEN Iris is *Virginica.*

3) IF *pw* is small AND *pl* is small AND *sw* is medium AND *sl* is small, THEN Iris is *Sestosa.*

Figure 3.19 illustrates the final output of the eTS structure incorporating the RLM-RLSE hybrid training technique. The amount of testing errors accumulated to 7. The recognition rate, based on 75 instances for verification, is 90.67%.

Figures 3.20 and 3.21 illustrate the results of the EFS clustering algorithm. Again, the rules are represented in a 2-dimensional input space, to simplify demonstration. The number of rules generated during the first random 75 instances of iris data is three; one less than the eTS scheme. The rule evolution based on the input data can be depicted in Figure 3.22. The EFS scheme did a better job at identifying the optimal amount of clusters needed for iris classification.

The major reason for the better identification is the rule-base modification approach introduced in step 6 of the algorithm. As depicted in Figure 3.22, when two clusters become compatible, in addition to being consistent, the rules are merged into one. This phenomenon can be seen at time instances 5 and 32.

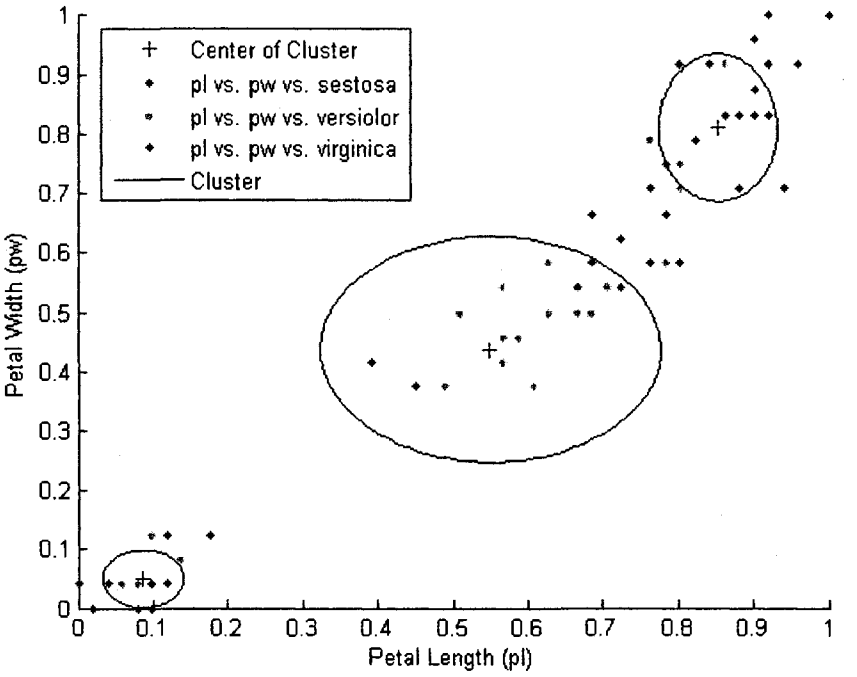

Figure 3.20. Petal Length vs. Petal Width for the EFS Scheme.

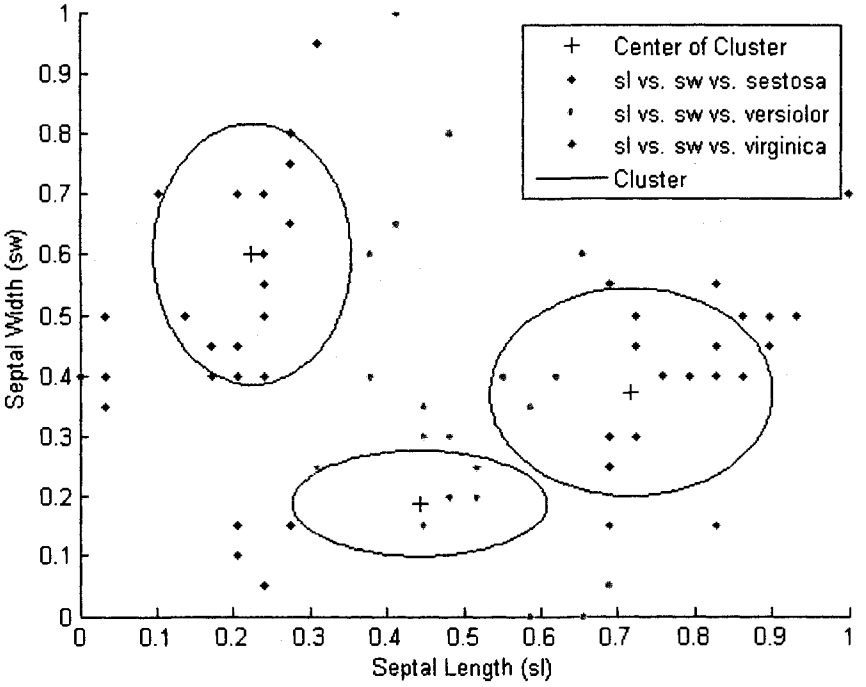

Figure 3.21. Sepal Length vs. Sepal Width for the EFS Scheme.

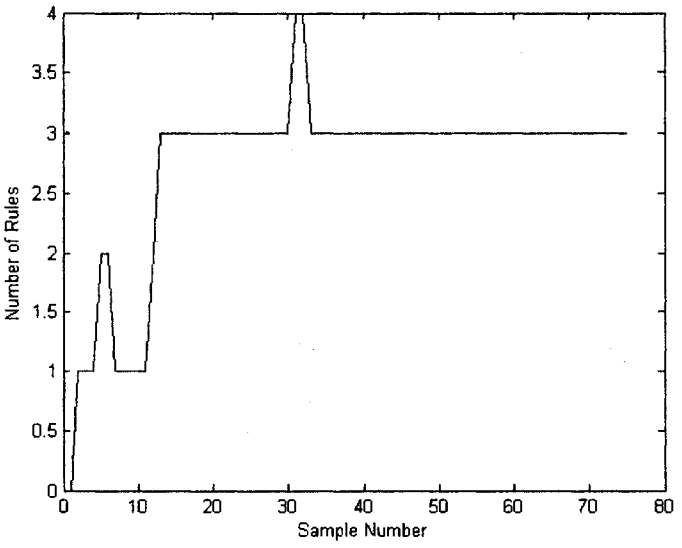

Figure 3.22. Evolving rule-base for the EFS Scheme.

The initial (before parameter optimization) MFs corresponding to the individual inputs are depicted in Figure 3.23. Next, the aforementioned structure will be subjected to GD-RLSE and RLM-RLSE parameter training.

Figure 3.24 shows the recognized structure, through GD-RLSE, and the corresponding membership functions of each feature after the parameter learning phase and 75 random instances of iris data. One hundred epochs were used for parameter training. The fuzzy rules of the iris classification can be extracted from the learned EFS structure from Figure 3.24 as follows.

1) IF *pw* is small AND *pl* is large AND *sw* is small AND *sl* is small, THEN Iris is *Versicolor.*

2) IF *pw* is medium AND *pl* is small AND *sw* is medium AND *sl* is medium, THEN Iris is *Virginica.*

3) IF *pw* is large AND *pl* is medium AND *sw* is large AND *sl* is large, THEN Iris is *Sestosa.*

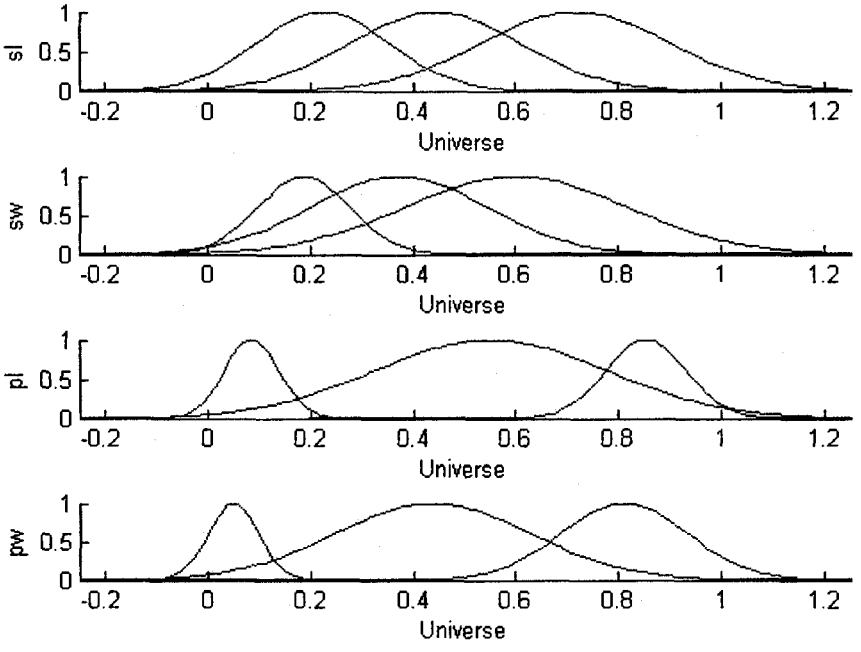

Figure 3.23. Initial MFs for the EFS Scheme.

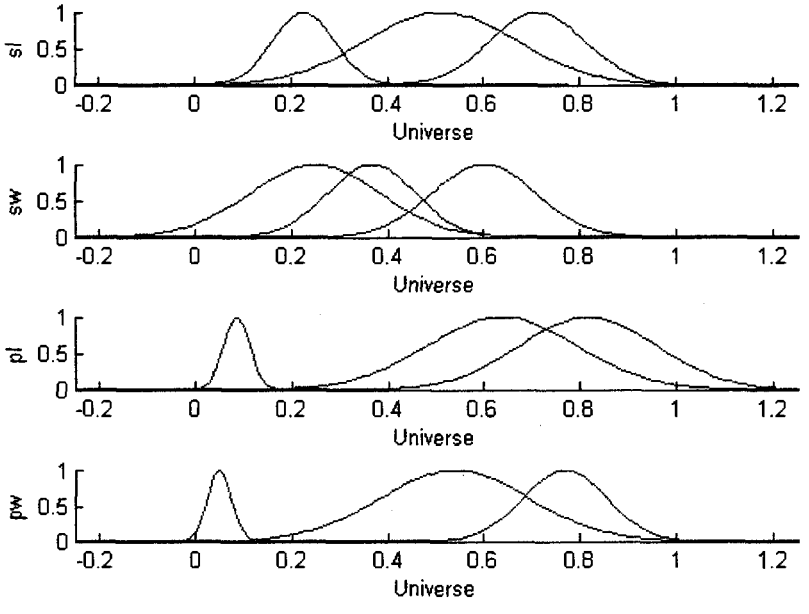

Figure 3.24. Final MFs for the EFS Scheme via GD-RSLE.

Figure 3.25 illustrates the final output of the EFS structure incorporating the GD-RLSE hybrid training algorithm. The amount of testing errors accumulated to 4. The recognition rate, based on 75 instances for verification, is 94.67%.

Figure 3.26 shows the identified structure, through RLM-RLSE, and the corresponding membership functions of each feature after the parameter learning phase and 75 random instances of iris data. One hundred epochs were used for parameter training. Within the training process, as in the case of GD-RLSE, some of the MFs amalgamated. The fuzzy rules of the iris classification can be extracted from the learned eTS structure from Figure 3.26 as follows.

1) IF *pw* is large AND *pl* is large AND *sw* is medium AND *sl* is small, THEN Iris is *Versiolor.*

2) IF *pw* is small AND *pl* is small AND *sw* is large AND *sl* is medium, THEN Iris is *Virginica.*

3) IF *pw* is small AND *pl* is large AND *sw* is large AND *sl* is large, THEN Iris is *Sestosa.*

Figure 3.27 illustrates the final output of the eTS structure incorporating the RLM-RLSE hybrid technique. The amount of testing errors accumulated to 3. The recognition rate, based on 75 instances for verification, is 96%.

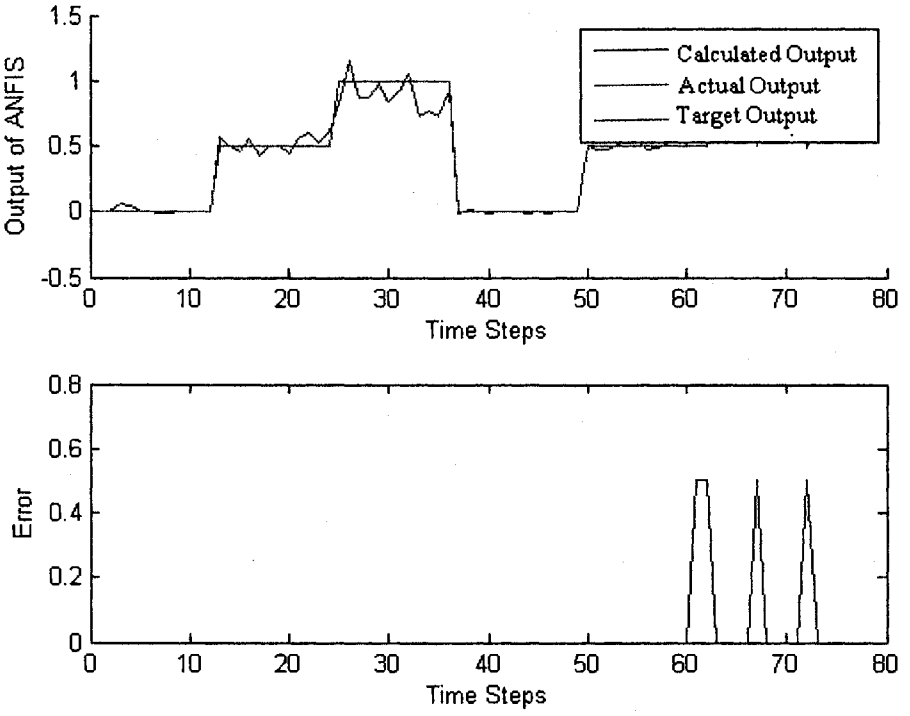

Figure 3.25. Final output for the EFS Scheme via GD-RSLE.

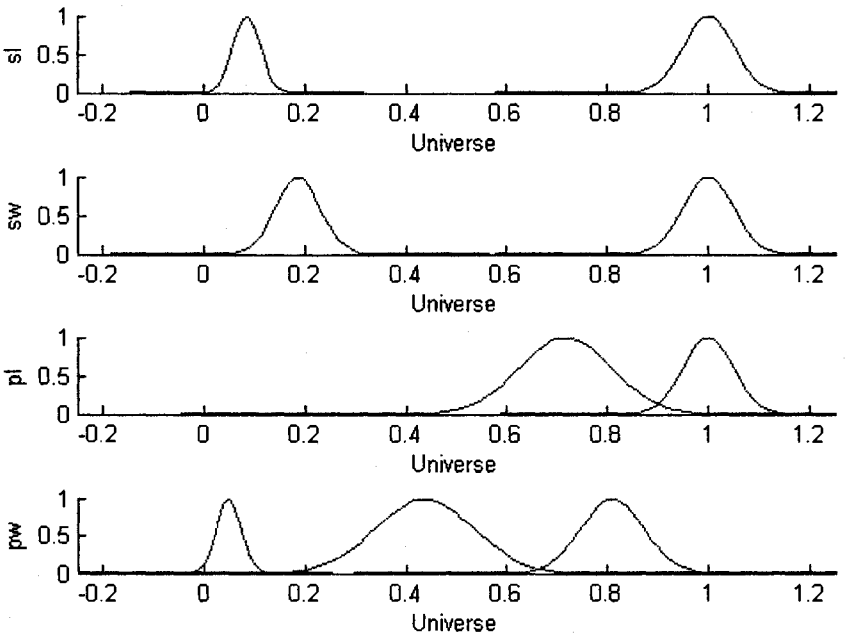

Figure 3.26. Final MFs for the EFS Scheme via RLM-RSLE.

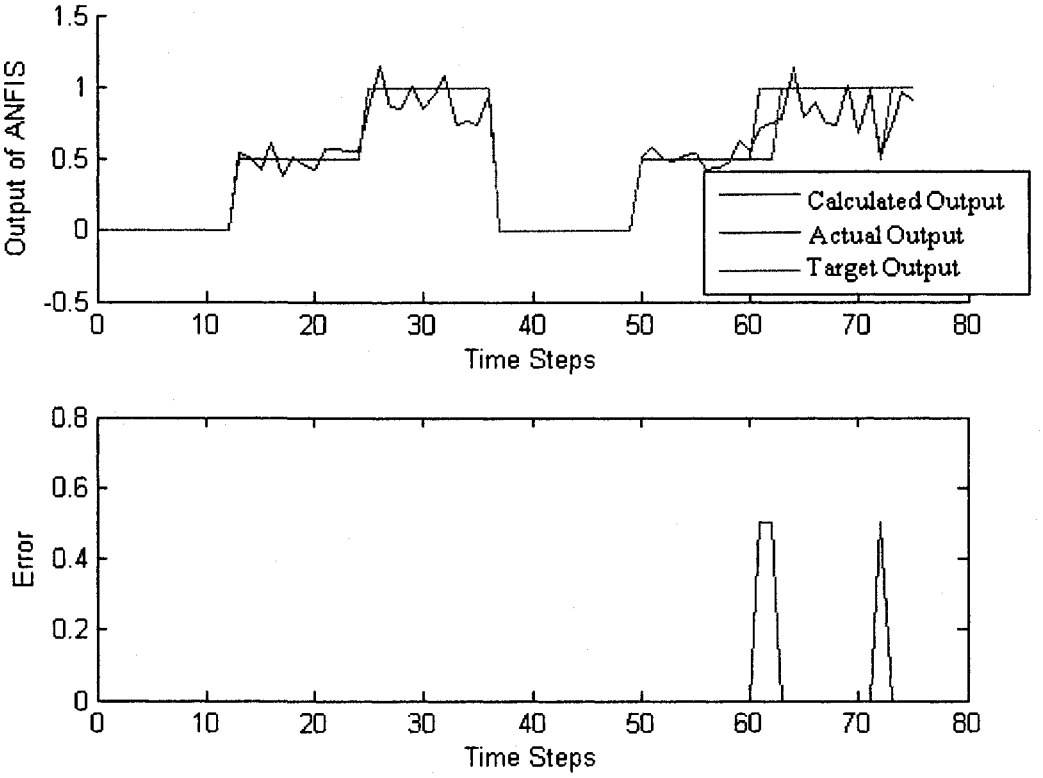

Figure 3.27. Final output for the EFS via RLM-RSLE.

3.4 Summary of the Results

Within this chapter, a novel approach to real-time and off-line clustering and prediction was introduced. The novel design incorporated simultaneous clustering of the input and output data sets, with the added constraint of mapping consistency and cluster compatibility. It was found that these constraints proved to help the EFS algorithm generate the optimal number of rules for clustering. As a result, the EFS algorithm outperformed the eTS potential-based algorithm in both real-time and off-line applications. A summary of the results can be found in Table 3.1.

Table 3.1. Results for off-line training of the eTS and EFS structures.

Structure	Training Technique	Number of Rules Generated	Epochs	Training Time (s)	Training Errors	Testing Errors	Recognition Rate
eTS	GD - RLSE	4	100	0.8	24	16	78.67%
	RLM - RLSE	4	100	1.54	4	7	90.67%
EFS	GD - RLSE	3	100	1.442	4	4	94.67%
	RLM - RLSE	3	100	2.684	1	3	96%

The EFS scheme generated fewer rules, training errors and testing errors than the potential-based eTS scheme. Also, the RLM-RLSE method proves to be a more powerful training algorithm than the classical GD-RLSE training algorithm. Albeit the EFS structure is more accurate in recognizing class, the training times for the EFS are slightly greater than that of the potential-based eTS. As mentioned in [8], however, the difference in the iteration time is considered insignificant for most real-time applications.

Figures 3.28 and 3.29 illustrate the resulting architecture of the eTS and EFS reasoning schemes, respectively.

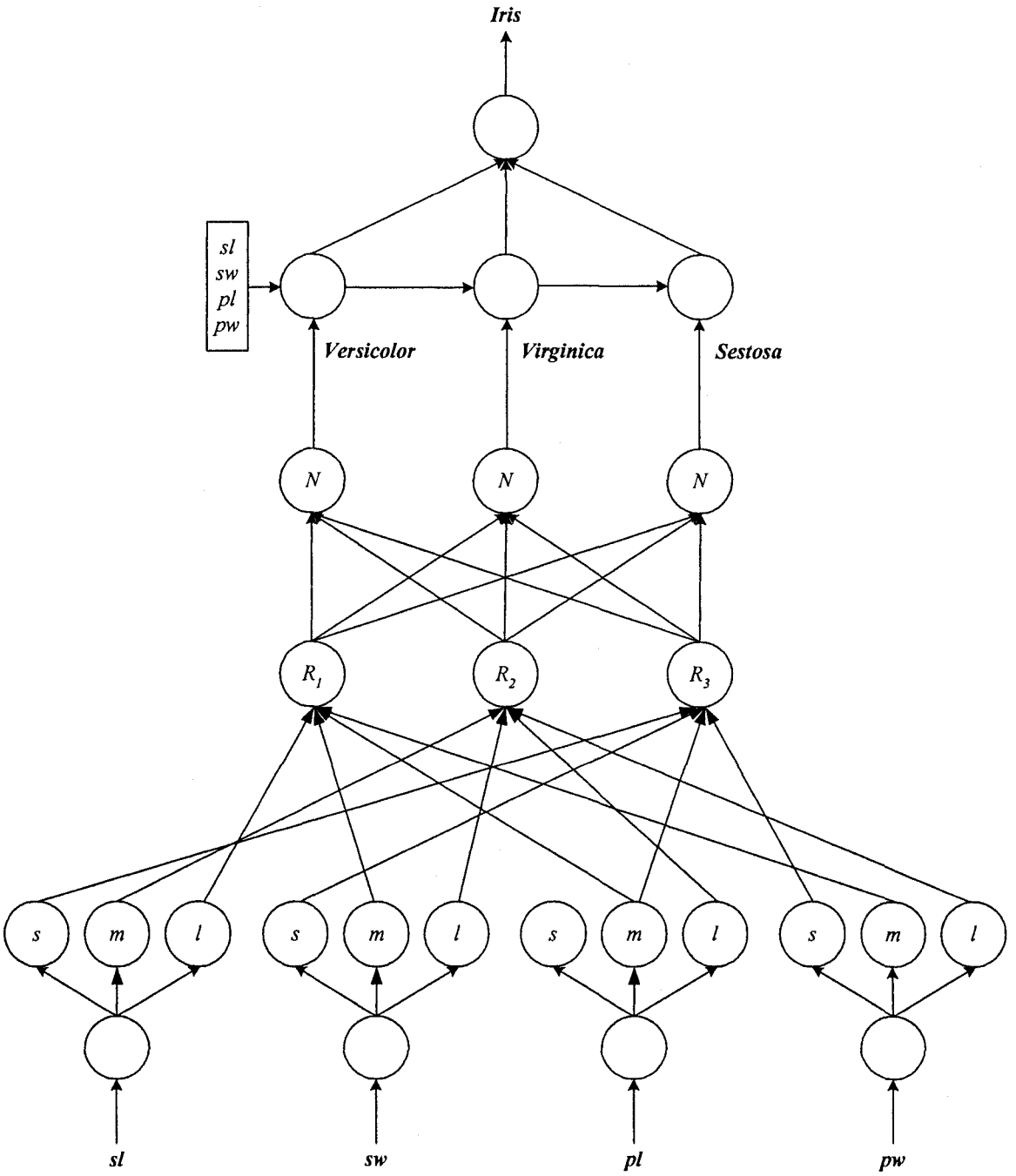

Figure 3.28. Architecture of the eTS via RLM-RLSE

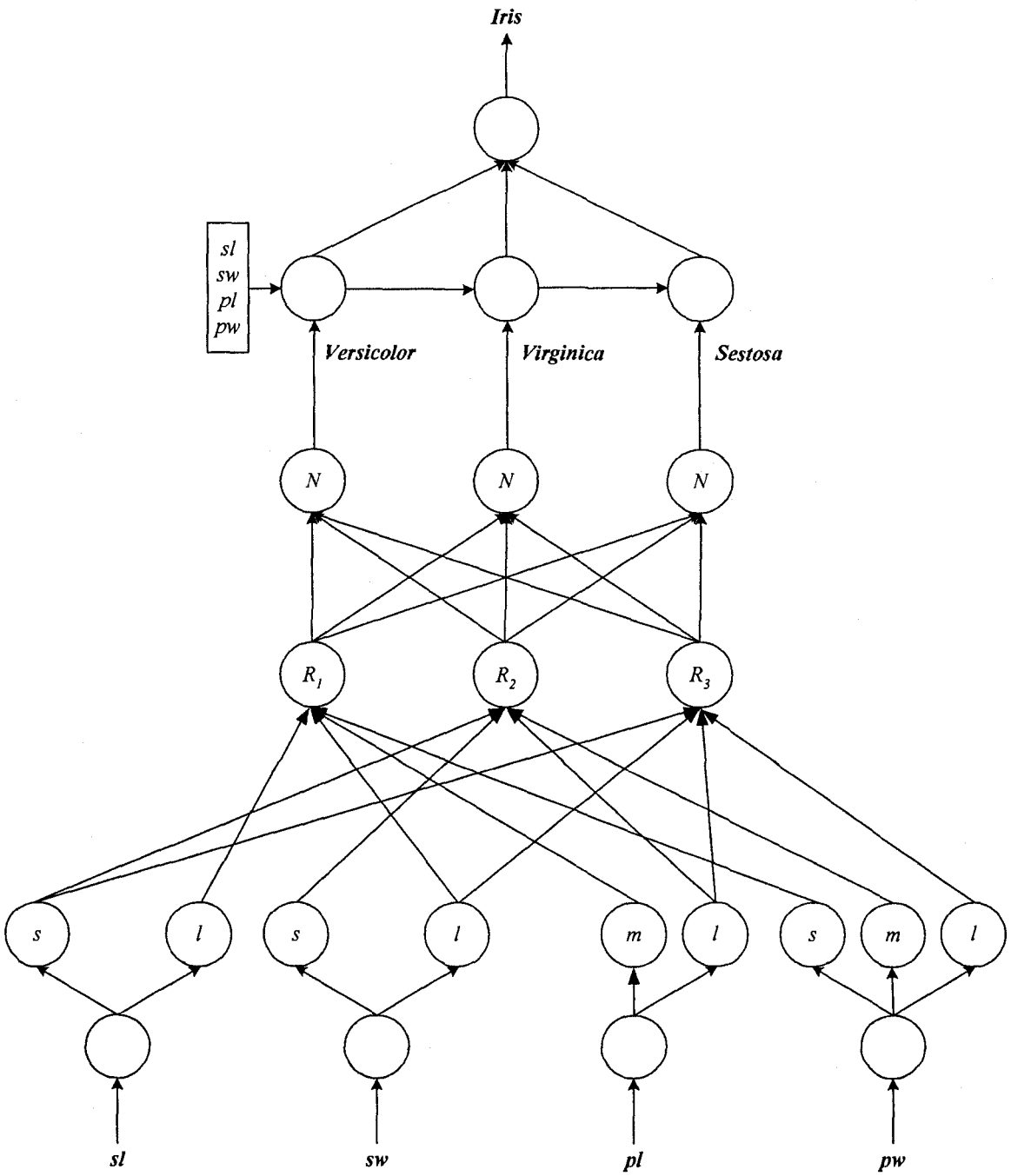

Figure 3.29. Architecture of the EFS via RLM-RLSE

In the next chapter, another novel approach to clustering and training is introduced. The new clustering scheme is a combination of the potential-based and EFS reasoning schemes. The new learning technique is a combination of the decoupled-extended Kalman filter [42, 43] and the RLSE.

Chapter 4

AN ENHANCED ETS (E^2TS) SCHEME FOR CLASSIFICATION/FORECASTING APPLICATIONS

In this chapter, a novel clustering technique, the enhanced-eTS (e^2TS), is developed for classification and forecasting applications. The e^2TS clustering technique is an enhanced version of the potential-based eTS clustering method as illustrated in Chapter 3. A novel training technique based on the integration of the decoupled-extended Kalman filter and RLSE is adopted for system training. The effectiveness of this new e^2TS technique is verified by simulation tests. Its performance is compared to other related clustering schemes such as the generic potential-based clustering technique, a transductive NF inference system with weighted data normalization and an NF inference system based on prior knowledge.

4.1Development of the e^2TS

4.1.1 Architecture of the e^2TS

The fuzzy model adapted here has its roots in the pioneering papers of Sugeno and his coworkers [44, 45] and is associated with the so-called Takagi-Sugeno fuzzy model – a special group of rule-base models with fuzzy antecedents and functional consequents that follow from the Takagi-Sugeno-Kang reasoning method [8].

If we consider an input vector $x = \{x_1, x_2, \cdots, x_n\}$ where n represents the number of inputs, which are normalized in a hypercube [0, 1], the fuzzy reasoning is performed by the following general representation:

$$\Re_j: \textbf{IF } (x_1 \text{ is } A_1^j) \textbf{ AND } \ldots \textbf{ AND } (x_n \text{ is } A_n^j) \textbf{ THEN}$$

$$(y_j = f_l^j = b_{0l}^j + b_{1l}^j x_1 + \cdots + b_{nl}^j x_n) \tag{4.1}$$

where $\Re_j$ denotes the j^{th} fuzzy rule, $j \in [1, N_r]$, N_r is the total number of fuzzy rules (or clusters); $b_\bullet^j$ are the consequent parameters; A_i^j is the j^{th} fuzzy set for x_i, $i \in [1, n]$; $y_j = [y_{j1}, y_{j2}, \cdots, y_{jM}]$ is an M-dimensional consequent (output) structure.

The e^{2}TS reasoning system is capable of taking on three different consequent reasoning structures: Type I (Mamdani model), Type II (zero order TS model) and the Type III (first order TS model), as discussed in Chapter 1. Due to the fact that Type III model are more accurate for non-linear mapping, the e^{2}TS scheme will adopt Type III reasoning as an example in this section for illustration.

In a NF inference system, nodes of the same layer usually have similar functions. In the proposed e^{2}TS, for example, all the fuzzy set MFs are in a Gaussian form

$$\mu_{A_i^j} = \exp\left(-\frac{(x_i - m_{ij})^2}{2\sigma_{ij}^2}\right) \tag{4.2}$$

where m_{ij} and σ_{ij} are the centers and spreads of the MFs, respectively. The Gaussian bell function has the characteristics of continuity and generalization property, and can be decomposed into multiple one-dimensional Gaussian MFs corresponding to different input variables. These properties can facilitate the implementation of input/output partitioning if each cluster is treated as a fuzzy rule (cluster) [49].

Despite the type of consequent reasoning structure selected, the premise rule structure remains the same. Assuming the use of a max-product operatior for the premise part of the structure, the rule firing strength is

$$\mu_j = \prod_{i=1}^{n}\left(\mu_{A_i^j}(x_i)\right) = \prod_{i=1}^{n}\exp\left(-\frac{(x_i - m_{ij})^2}{2\sigma_{ij}^2}\right) = \exp\left(-\sum_{i=1}^{n}\frac{(x_i - m_{ij})^2}{2\sigma_{ij}^2}\right), j \in [1, N_r]. \quad (4.3)$$

Once the firing strength is calculated via (4.3), the firing strengths are normalized. After normalization, the overall output will be

$$y = \sum_{j=1}^{N}\frac{\mu_j}{\mu_\Sigma} f_l^j, \quad j \in [1, N_r] \quad (4.4)$$

where f_l^j is the result from the consequent part (where $M = 1$ for classification); $\frac{\mu_j}{\mu_\Sigma}$ is the normalized firing strength of the j^{th} rule, in which

$$\mu_\Sigma = \sum_{j=1}^{N}\mu_j = \sum_{j=1}^{N}\exp\left(-\sum_{i=1}^{n}\frac{(x_i - m_{ij})^2}{2\sigma_{ij}^2}\right), \quad (4.5)$$

where μ_j is determined by (3).

4.1.2 The e^{2}TS Clustering Procedure

The suggested e^{2}TS is an enhanced version of the well known evolving eTS system [8]. The e^{2}TS is different from the standard eTS by use of a recursively adaptive updating algorithm of the spread and centers. Unlike the eTS system, the e^{2}TS recursively updates the cluster centers (and spreads) based on the volume of data of the corresponding cluster(s). In this way, the e^{2}TS system will generate a significantly less amount of rules [9]. Figure 4.1 illustrates the flowchart of the clustering algorithm of the suggested e^{2}TS. Each part of the process is described in full detail below.

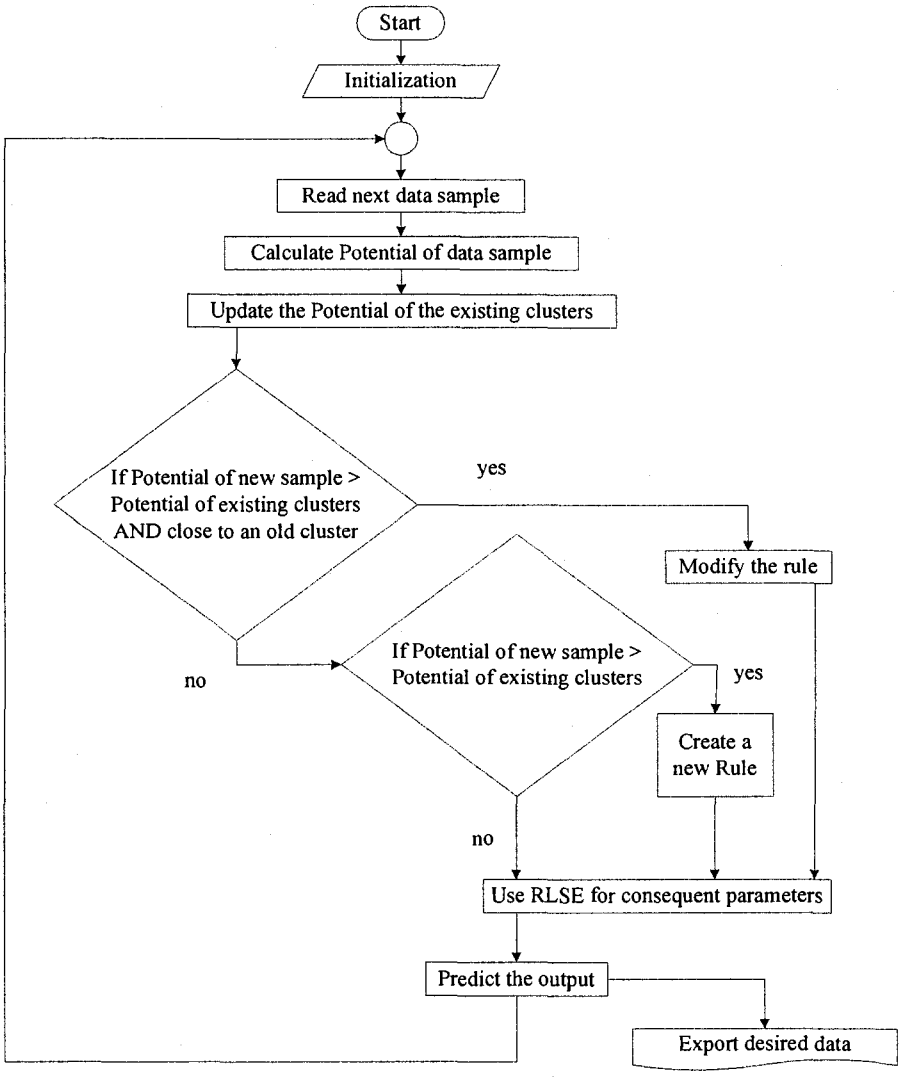

Figure 4.1. Flowchart for the e^{2}TS clustering processes.

Step 1: *Initialization.* The e^{2}TS starts with one rule. The first rule (cluster) is located at the first input data sample. The radius (spread) is initialized to 0.25. Then

$$N_r \leftarrow 1;\quad N_W^j \leftarrow 1;\quad k \leftarrow 1;\quad x_k \leftarrow x_{1n};\quad P_k(z_k) \leftarrow 1;\quad m_{ij} \leftarrow x_k;\quad \sigma_{ij} \leftarrow 0.25$$

where N_W^j is the number of samples in cluster j; $j \in [1, N_r]$ and j is initialized to 1; m_{ij} and σ_{ij} are the cluster centers and spreads, respectively; $z = [x; y]$; and $P_k(z_k)$ is the potential of data sample k.

Step 2: *Input the next data sample.*

$$k \leftarrow k+1$$

The new data sample z_k is collected.

Step 3: *Potential of the new data sample.* After collecting z_k, the potential of z_k is calculated as follows:

$$P_k(z_k) = \frac{k-1}{(k-1)(\vartheta_k+1)+\sigma_k-2\upsilon_k} \tag{4.6}$$

where $\vartheta_k = \sum_{j=1}^{n+1}(z_k^j)^2$; $\sigma_k = \sigma_{k-1} + \sum_{j=1}^{n+1}(z_{k-1}^j)^2$; $\upsilon_k = \sum_{j=1}^{n+1} z_k^j \beta_k^j$; and $\beta_k^j = \beta_{k-1}^j + z_{k-1}^j$ as in [8];

β_k^j and σ_k are initialized to the appropriate vector of zeros.

Step 4: *Potential update of existing clusters.* The recursive formula for updating the potential of all existing clusters is given as:

$$P_k(\overline{z}_j) = \frac{(k-1)P_{k-1}(\overline{z}_j)}{k-2+P_{k-1}(\overline{z}_j)+P_{k-1}(\overline{z}_j)\sum_{j=1}^{n+1}\left(d_{k(k-1)}^j\right)^2} \tag{4.7}$$

where $\overline{z}_j$ represents the x and y coordinates of all existing clusters; $P_k(\overline{z}_j)$ is the potential of all existing clusters; and $d_{k(k-1)}^j = z_k^j - z_{k-1}^j$. $\overline{z}_j$ is initialized to z_k and $P_k(\overline{z}_j)$ is initialized to 1 when $k \leftarrow 1$.

Step 5: *Structure recognition.* **IF** the potential of the current cluster is greater than the potential of all existing clusters **AND** the new data point is close to an old cluster, **THEN** merge the new cluster. If both conditions hold, merge the new data set to the existing cluster.

That is:

$$\textbf{IF } P_k(z_k) > P_k(\bar{z}_j) \textbf{ AND } \frac{P_k(z_k)}{\max_{j=1}^{N_r} P_k(\bar{z}_j)} - \frac{\delta_{\min}}{0.25} \geq 1. \tag{4.8}$$

THEN $N_r \leftarrow N_r$; and the volume of the cluster becomes: $N_W^l \leftarrow N_W^l + 1$ where $l = \arg\left(\max_{j=1}^{N_r} \left(P_k(\bar{z}_j) \right) \right)$. The new centers and spreads are recursively updated via the following algorithm:

$$\left(\boldsymbol{\sigma}_{l,k}\right)^2 \leftarrow \left(\boldsymbol{\sigma}_{l,k-1}\right)^2 + \frac{1}{N_W^l}\left[\left(\mathbf{x}_k - \mathbf{m}_{l,k-1}\right)^2 - \left(\boldsymbol{\sigma}_{l,k-1}\right)^2\right], \tag{4.9}$$

$$\mathbf{m}_{l,k} \leftarrow \mathbf{m}_{l,k} + \frac{\mathbf{x}_k - \mathbf{m}_{l,k}}{N_W^l}. \tag{4.10}$$

where $\mathbf{m}_{l,k}$ and $\boldsymbol{\sigma}_{l,k}$ represent the updated cluster centers and spreads, respectively.

IF, however, the potential of the current cluster is greater than the potential of all existing clusters **AND** the new data point is not close to an old cluster, **THEN** create a new cluster. That is

$$N_r \leftarrow N_r + 1;\ N_W^{N_r} \leftarrow 1;\ m_{iN_r} \leftarrow x_k;\ \text{and } \sigma_{iN_r} \leftarrow 0.25.$$

Step 6: *Tuning the consequent parameters via RLSE.* If the Type III or Type II structure is desired, train the consequent parameters, recursively, via the RLSE algorithm.

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} + \boldsymbol{\Psi}_k \boldsymbol{\Omega}_{k-1}(d_{k-1} - \boldsymbol{\Omega}_{k-1}^T \boldsymbol{\theta}_{k-1}), \tag{4.11}$$

$$\boldsymbol{\Psi}_k = \frac{1}{\alpha_{k-1}}\left[\boldsymbol{\Psi}_{k-1} - \boldsymbol{\Psi}_{k-1}\boldsymbol{\Omega}_{k-1}\left[\alpha_{k-1} + \boldsymbol{\Omega}_{k-1}^T \boldsymbol{\Psi}_{k-1}\boldsymbol{\Omega}_{k-1}\right]^{-1} \boldsymbol{\Omega}_{k-1}^T \boldsymbol{\Psi}_{k-1}\right], \tag{4.12}$$

where $k = 1, 2, \ldots, K$ and α_k is a forgetting factor in the range of $\alpha_k = [0.9, 0.99]$. The covariance matrix is initialized as $\boldsymbol{\Psi}_0 = \rho \boldsymbol{I}$, where $\boldsymbol{I}$ is an identity matrix and δ is a constant normally in the range of $\delta = [10^2, 10^4]$.

Step 7: *Output prediction:* At this stage, the output is predicted via

$$\hat{y}_{k+1} = \boldsymbol{\Omega}_k^{\mathrm{T}} \boldsymbol{\theta}_k \tag{4.13}$$

and compared to the desired value. The corresponding results can be exported for data analysis. When $k = K$, end the program and proceed to DEKF-RLSE training; otherwise proceed back to *step 2*.

4.2 Introduction to the DEKF-RLSE Hybrid Training Technique

Once the structure is established through the aforementioned clustering procedure, the premise and consequent parameters are optimized by a hybrid training technique: the decoupled extended Kalman filtering (DEKF) and the RLSE. The consequent parameters are linear and hence are optimized by the RLSE algorithm in the forward pass. The premise parameters are nonlinear and will be optimized by the DEFK procedure in the backward pass.

The derivation of the DEKF is based upon a natural simplification of the global extended Kalman algorithm (GEK) of Singhal and Wu [46] by ignoring the interdependence of mutually exclusive groups of weights [42]. The DEKF is based upon the assertion that the weights of the network can be grouped (or decoupled) such that the elements of $\mathbf{P}_i(k)$ corresponding to weights from different groups can be ignored [43]. Accordingly, if the weight groups are judicially chosen (e.g. by node, layer or subnetwork) then a sizeable reduction in training time is found.

Given a network with M weights and N_L output nodes, partition the weights into g groups, with m_i weights in group i. The weight update equations at time step k of the DEKF is given by:

$$\mathbf{v}_i(k) = \mathbf{P}_i(k)\mathbf{u}_i(k) \tag{4.14}$$

$$\mathbf{A}(k) = \left[\mathbf{R}(k) + \sum_{i=1}^{G}\left\{\mathbf{u}_i^T(k)\mathbf{v}_i(k)\right\}\left[\varphi_i(k)\varphi_i^T(k)\right]\right]^{-1} \tag{4.15}$$

$$\hat{\theta}(k+1) = \hat{\theta}(k) + \left\{\varphi_i^T(k)(\mathbf{A}(k)\xi(k))\right\}\mathbf{v}_i(k) \tag{4.16}$$

$$\mathbf{P}_i(k+1) = \mathbf{P}_i(k) - \left\{\varphi_i^T(k)(\mathbf{A}(k)\varphi_i(k))\right\}\left[\mathbf{v}_i(k)\mathbf{v}_i^T(k)\right] \tag{4.17}$$

where $\mathbf{v}_i(k)$ is a m_i-by-N_L matrix containing the Kalman gains for weight group i. $\mathbf{P}_i(k)$ denotes a m_i -by- m_i matrix defined as the approximate conditional error covariance matrix for group i. $\mathbf{u}_i(k)$ is the input vector for the i^{th} node. $\mathbf{A}(k)$ is non-singular and a N_L-by-N_L matrix that is referred to as the global scaling matrix. $\mathbf{R}(k)$ represents a diagonal N_L-by-N_L matrix whose diagonal components are equal to or slightly less than 1. $\varphi_i(k)$ is a vector of partial derivatives of the network's output node signals with respect to the i^{th} node's net input and $\hat{\theta}(k)$ is a vector of length m_i containing the weight values of group i.

Two natural members exist in this set of DEKF algorithms. They are referred to as *node-decoupled* EKF or NDEFK, and the limiting case in which each weight of the network constitutes a group in itself, which is referred to as *fully-decoupled* EKF or FDEKF.

Based upon group partitioning, the computational complexity of the NDEKF is reduced from $O(N_L^2 M + N_L \sum_{i=1}^{G} m_i^2)$ to $O(N_L^2 G + N_L \sum_{i=1}^{G} m_i^2)$ whereas the computational complexity of the FDEKF is reduced from $O(N_L^2 M + N_L M)$ to $O(N_L^2 G + M)$ [43]. As a result, to save time in training, the FDEKF will be adapted in this chapter.

Due to a lack of a forcing function (see Reference [4]) in [43], the DEKF can lead to some computational difficulties. It is possible that $\mathbf{P}(k)$ can become singular, lose the necessary property of nonnegative definiteness, or vanish completely [43]. In order to prevent such difficulties, an artificial noise process will be introduced into equation (4.17) which then becomes

$$\mathbf{P}_i(k+1) = \mathbf{P}_i(k) - \left\{\varphi_i^T(k)(\mathbf{A}(k)\varphi_i(k))\right\}\left[\mathbf{v}_i(k)\mathbf{v}_i^T(k)\right] + \mathbf{Q}_i(k) \tag{4.18}$$

where $\mathbf{Q}_i(k)$ is a diagonal matrix with small non-negative components in the range $[10^{-6} \quad 10^{-2}]$.

4.3 Performance Evaluation

The performance of the proposed e^{2}TS is compared to the generic potential-based eTS clustering technique [8], a transductive neuro-fuzzy inference system with weighted data normalization (TWNFIS) [47] and a NF inference system based on prior knowledge. The above four techniques are compared by use of the well accepted GD – RLSE hybrid training technique.

To compare the above structures, the Wisconsin Breast Cancer Diagnostic Data is used.

Six hundred eighty three patterns are used to evaluate the performance of the proposed structure. Each data sample consists of 9 features: clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli and mitoses. The data is normalized to the range [0, 1] and is divided as follows: 250 data samples are used for structure recognition, 230 are used for premise and consequent parameter optimization and 203 data samples are used for verification.

The output is classified as follows

$$Class = \begin{cases} Benign, & if\ y < 0.5 \\ Malignant, & if\ 0.5 \le y \end{cases} . \tag{4.19}$$

Table 4.1, summarizes the results of the experiment. The results located in the Table are an average over 10 trials to minimize random effects related to initial values of weights. The TWNFIS created more clusters and took the most time in training than the other systems used for comparison, but did, however, outperform the custom NF and eTS structures that had 94.1% and 96.6% recognition rates, respectively. The custom NF system is constructed based on prior knowledge. Four static rules have been given to the custom design and were based on the rule generation of the eTS model. The eTS model did outperform the custom NF design (96.6% to 94.1%), but took more time in training. This is due to the fact that the eTS is generating its own rule-base whereas the custom NF design already had one established.

Table 4.1. Results of the GD-RLSE Comparison.

Neuro-Fuzzy Models	No. of Rules	Epochs	Training time (avg.)	Training Errors (avg.)	Testing Errors (avg.)	Recognition Rate (%)
custom neuro-fuzzy	4	100	21.0211	3	7	94.1
TWNFIS	5	100	29.77639	1.2	2.8	97.2
eTS	4	100	23.0211	0.9	3.4	96.6
e^2TS	2	100	1.5632	0.09	0.2	97.8

Due to the recursively updating of the radius and spread in the e^2TS structure, the e^2TS generated less rules and hence less time for training. Also, the e^2TS outperformed all the other mentioned structures.

The performance of the DEKF-RLSE hybrid training technique is compared to the GD-RLSE and the RLM-RLSE hybrid algorithms via the novel structure identification technique constructed in this paper. Analogous to the first part in this evaluation, the mentioned hybrid training techniques are compared using the same Wisconsin Breast Cancer Diagnostic data series.

Table 4.2. Results for the training techniques.

Training Method	Training time per epoch (avg.)	Training Errors (avg.)	Testing Errors (avg.)	Recognition Rate (%)
GD - RLSE	1.5632 s	0.12	2.2	97.8
RLM - RLSE	1.7225 s	0.09	1	99
DEKF - RLSE	1.8076 s	0.07	0.2	99.8

It can be seen, in Table 4.2, that the DEKF-RLSE hybrid technique outperforms the other hybrid algorithms in terms of recognition rate, training errors and testing errors. Due to the highly complex calculations in the DEKF [43], however, the DEKF-RLSE technique takes more amount of time to train.

Since processing power in computers is increasing[4], the added time is insignificant; especially for application specific problems [4].

Figure 4.2 illustrates the generated MFs of the e^{2}TS after clustering. Figure 4.3 depicts the learned MFs of the e^{2}TS via the DEKF-RLSE through 100 epochs. The final structure is architecturally shown in Figure 4.4.

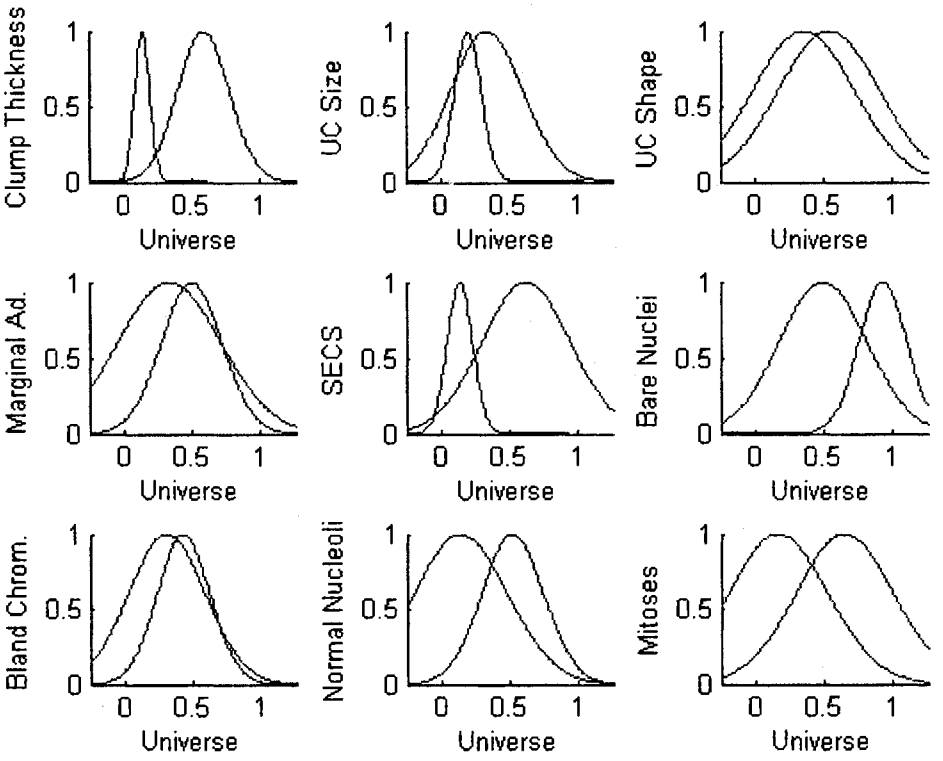

Figure 4.2. MFs of the e^{2}TS before parameter training.

[4] See Moore's Law.

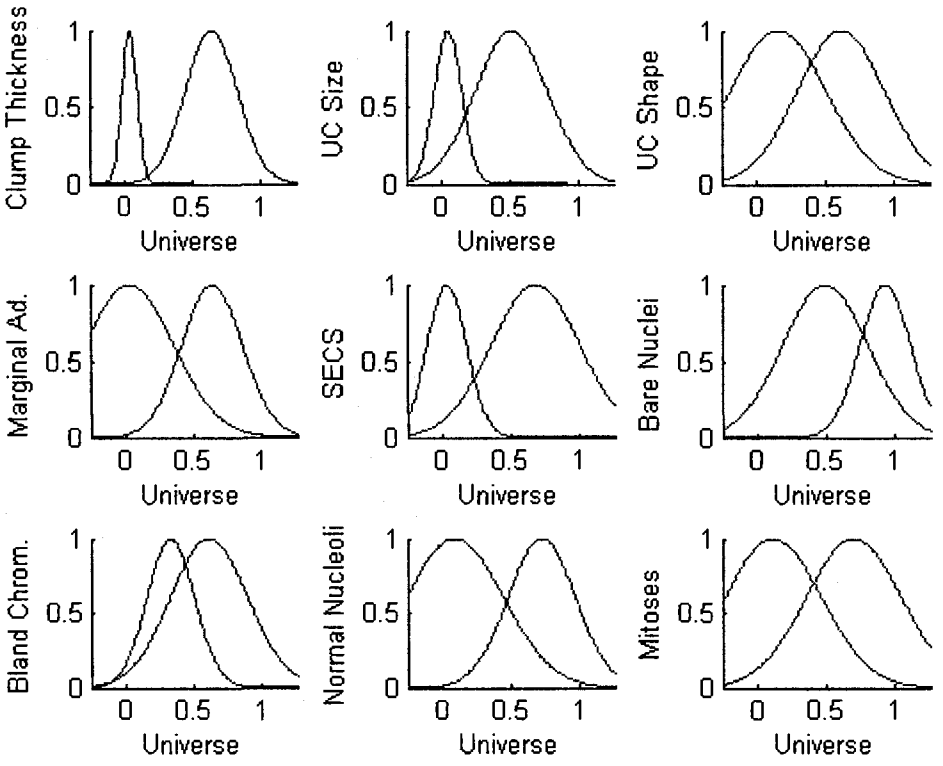

Figure 4.3. MFs of the e^2TS after parameter training.

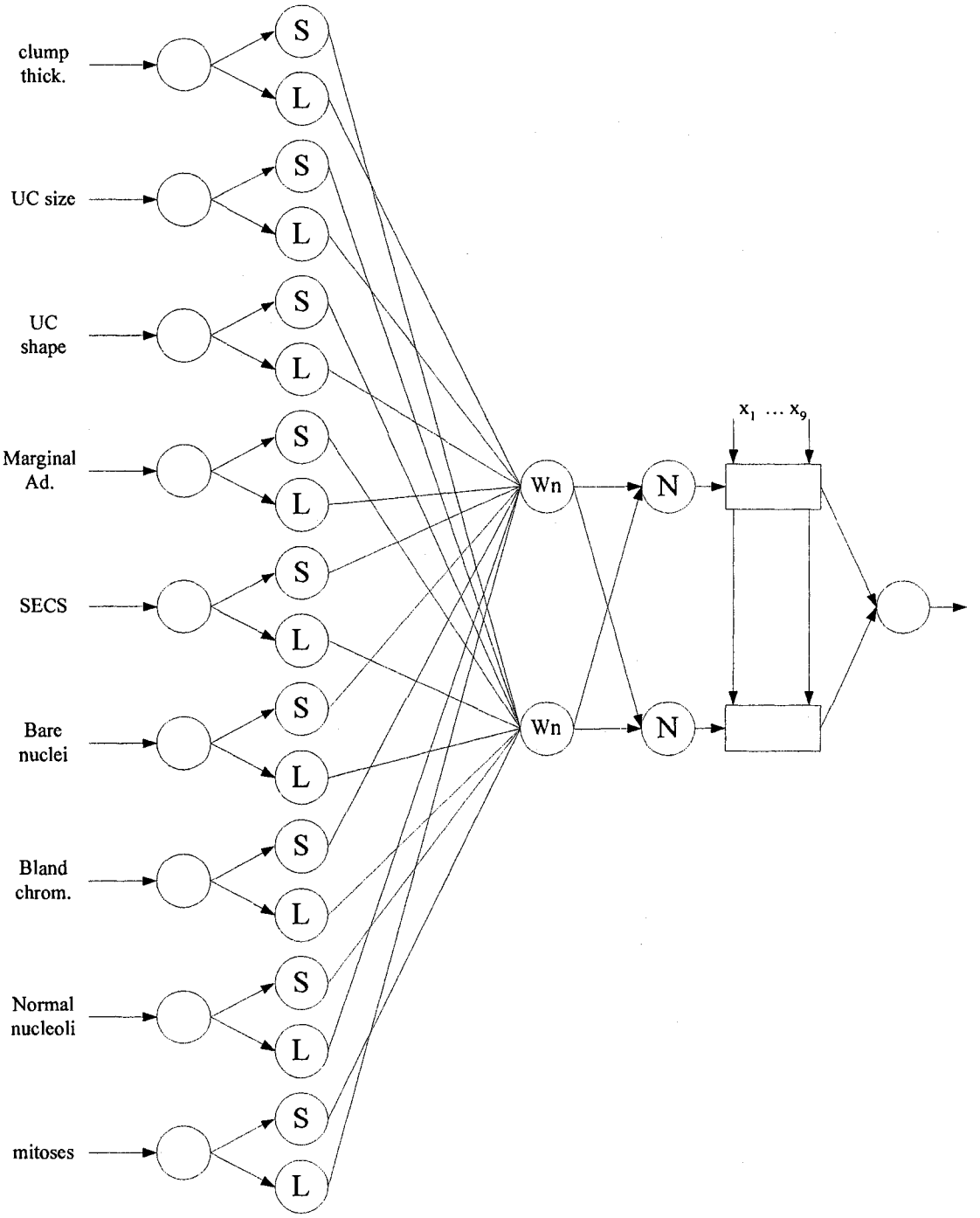

Figure 4.4. Architecture of the e2TS after the application of DEKF-RLSE.

4.4 Summary of the Results

In this chapter a novel clustering algorithm and training technique were proposed. The new e^2TS technique is an enhanced version of the potential-based evolving TS clustering algorithm. From experimental results, it is proven that the new e^2TS outperforms the generic potential-based eTS clustering technique [8], a transductive NF inference system with weighted data normalization [47] and a NF inference system based on priori knowledge in terms of training errors, testing errors, recognition rate and training time. Test results have shown that the new scheme can generate fewer amounts of rules but higher accuracy than the other techniques.

The new training technique, the DEKF-RLSE, also proved to be a superior training technique compared to the other related classical algorithms. The DEKF-RLSE outperformed the others in terms of training errors and testing errors, albeit it took a little more time in training operations.

Chapter 5

CONCLUSIONS AND FUTURE WORK

5.1 Summary of the Main Results

The objective of this thesis was to develop more accurate and robust multi-step ahead predictors and classifiers for industrial applications. In Chapter 2, step-inputs were compared to sequential-step inputs via the feed-forward NN, recurrent NN and the proposed weighted NF inference system. Through experimental results, it was proven that an equal-step-based predictor outperformed a sequential-step counterpart through many different time-steps. Also in Chapter 2, experimental results showed that developed weighted recurrent NF inference system outperformed the feed-forward NN and recurrent NN in both sequential-based and equal-step-based input situations. Albeit the results in Chapter 2 were promising, there still loomed one problem for some applications. Without any prior information pertaining to the number of rules for successful mapping, it is hard to construct a fixed structure for classification and prediction. This problem led to the investigation of self-organizing or evolving NF paradigms.

In Chapter 3, a novel approach to clustering was suggested. In the new scheme, both input and output patterns were clustered simultaneously, with the added constraints of mapping consistency and cluster compatibility.

These extra constraints eliminated clusters forming due to noise affected data. Ergo, more meaningful clusters were formed. The new technique was compared to the related classical paradigms such as the potential-based eTS clustering algorithm. The new method outperformed the eTS algorithm in terms of accuracy and number of rules generated, albeit the new algorithm did take a marginal time longer to develop its structure due to the extra constraints. Furthermore, a novel hybrid training algorithm based on the recursive LM and recursive LSE was adopted for system training. Simulation test results demonstrated that the new algorithm outperformed the well-accepted potential-based eTS schemes.

In Chapter 4 an enhancement to the potential-based clustering technique was suggested. The so-called e^2TS clustering algorithm recursively updated the cluster centers (and spreads) based on the volume of data of the corresponding cluster(s). This enhancement allowed the e^2TS to generate fewer clusters then its potential-based cousin. The e^2TS was rigorously simulated against other clustering techniques. Through a performance evaluation it was found that the e^2TS scheme outperformed all the related classical schemes. Furthermore, another novel training algorithm based on the decoupled extended Kalman filter and recursive LSE was suggested. This technique is a hybrid technique, in the sense that the RLSE is performed in the forward pass (the consequent parameters) and the DEKF is performed in the backward pass (the premise parameters). Rigorous simulations verified the effectiveness of the adopted novel training technique.

All the suggested techniques and tools will have a great potential to be used directly or indirectly for different types of applications, such as machinery condition monitoring, fault diagnostics and prognostics, predictive control, bioinformatics, earthquake forecasting, to name but a few.

5.2 Future Work

Future work to be carried out is summarized as follows:

1) Implement the developed e^2ST scheme for real applications, such as machinery fault diagnostics, system state forecasting, and material property prediction, and gene production analysis.
2) Develop novel forecasting and online training strategies for predictive (functional) control in parallel robots.
3) Suggest more effective training algorithms to further improve training speed and convergence related to local minima.

References

[1] Hussein A. Abbass. An evolutionary artificial neural networks approach for breast cancer diagnosis. Artificial Intelligence in Medicine. 2002.

[2] Zadeh, L.A. (1965). Fuzzy sets. Information and Control Vol. 8, No. 3, pp. 338-353.

[3] J. S. R. Jang, C. T. Sun and E. Mizutani. Neuro-Fuzzy and Soft Computing. Prentice Hall, 1997.

[4] W. Wang. An Intelligent System for Machinery Condition Monitoring. IEEE Transactions on Fuzzy Systems (in press).

[5] R. Bone and M. Crucianu. Multi-step-ahead Prediction with Neural Networks: a Review. Approaches Connexionnistes en Sciences Economiques et en Gestion, 21-22 November 2002, Boulogne sur Mer, France. pp. 97-106.

[6] M.Mackey, L. Glass. Oscillations and Chaos in Physiological Control Systems. Science, 1997, pp.197-287.

[7] Plamen Angelov and Richard Buswell. Evolving Rule-based Models: a Tool for Intelligent Adaptation. Heidelberg, Germany: Springer-Verlag, 2002.

[8] Plamen P. Angelov and Dimitar P. Filev. An Approach to Online Identification of Takagi-Sugeno Fuzzy Models. IEEE Transactions on Systems, Man and Cybernetics Part B: Cybernetics, Vol. 34, No.1, February 2004. pp. 484-498.

[9] Angelov, P., X. Zhou. Evolving Fuzzy Systems from Data Streams in Real-Time. 2006 International Symposium on Evolving Fuzzy Systems, September 2006, pp. 29-35.

[10] J. S. Wang and C. S. G. Lee. Self-Adaptive Neural-Fuzzy Inference Systems for Classification Applications. IEEE Transactions on Fuzzy Systems, Vol. 10, No. 6, December 2002, pp. 790-802.

[11] H. M. Lee, C. M. Chen, J. M. Chen and Y. L. Jou. An Efficient Fuzzy Classifier with Feature Selection Based on Fuzzy Entropy. IEEE Transactions Systems, Man, Cybernetics Part B: Cybernetics, Vol. 31, June 2001, pp. 426-432.

[12] H. M. Lee. A Neural Network Classifier with Disjunctive Fuzzy Information. Neural Networks, Vol. 11, No. 6, 1998, pp. 1113-1125.

[13] P. K. Simpson. Fuzzy min-max Neural Networks-Part I: Classification. IEEE Transactions on Neural Networks, Vol. 3, September 1992, pp. 776-786.

[14] T. P. Wu and S. M. Chen. A New Method for Constructing Membership Functions and Fuzzy Rules From Training Examples. IEEE Transactions Systems Man, Cybernetics Part B, Vol. 29, February 1999, pp. 25-40.

[15] M. Pourahmadi. Foundations of Time Series Analysis and Prediction Theory, John Wiley & Sons Inc., 2001.

[16] D. Chelidze and J. Cusumano. A Dynamical Systems Approach to Failure Prognosis, Journal of Vibrations and Acoustics, 2004, Vol. 126, pp 1-7.

[17] C. Li and H. Lee, Gear Fatigue Crack Prognosis Using Embedded Model, Gear Dynamic Model and Fracture Mechanics, Mechanical Systems and Signal Processing, Vol. 9, pp 836-846.

[18] J. Connor, R. Martin and L. Atlas. Recurrent Neural Networks and Robust Time Series Prediction. IEEE Transactions on Neural Networks Vol. 5, 1994, pp 240-254.

[19] S. Y. Liang and X. Liang. 2006. Improving Signal Prediction Performance of Neural Networks Through Multi-Resolution Learning Approach. IEEE Transactions on Systems Man and Cybernetics Part B, Vol. 36, 341-352.

[20] D. Husmeier. 1999. Neural Networks for Conditional Probability Estimation: Forecasting beyond Point Prediction, Springer-Verlag London Ltd.

[21] J. Jang, ANFIS: Adaptive-Network-Based Fuzzy Inference System. IEEE Transactions on Systems, Man and Cybernetics Part B, Vol. 23, 1993, pp. 665-685.

[22] W. Wang, An Adaptive Predictor for Dynamic System Forecasting. Mechanical Systems and Signal Processing, 2007, Vol. 21, pp. 809-823.

[23] W. Wang, F. Ismail, and F. Golnaraghi. A Neuro-Fuzzy Approach for Gear System Monitoring, 2004, IEEE Trans. Fuzzy Syst. 12: 710-723.

[24] H. Demuth, M. Beale and M. Hoga. Neural Network Toolbox: User's guide. Version 5. Mathworks Inc. 2007.

[25] J. Vrbanek, Jr. and W. Wang. Data-Driven Forecasting Schemes: Evaluations and Applications. Journal of Computer Science, 2007, Vol. 9, 747-753.

[26] J. Jang, ANFIS: Adaptive-Network-Based Fuzzy Inference System. IEEE Transactions on System Man and Cybernetics Part B, 1993, Vol. 23, pp. 665-685.

[27] M. Figueiredo, R. Ballini, S. Soares, M. Andrade, and F. Gomide, Learning Algorithms for a Class of Neuro-Fuzzy Network and Applications, IEEE Transactions on Systems Man and Cybernetics - Part C, 2004, Vol. 34, pp. 293-301.

[28] H. Ishibuchi and T. Yamamoto, Rule Weight Specification in Fuzzy Rule-Based Classification Systems. IEEE Transactions on Fuzzy Systems, 2005, Vol. 13, pp. 428-435.

[29] D. Nauck, 2000. Adaptive Rule Weights in Neuro-Fuzzy Systems, Journal of Neural Computation and Applications, Vol. 9, pp. 60-70.

[30] K. Tanka, M. Sano and H. Watanabe, Modeling and Control of Carbon Monoxide Concentration Using a Neuro-Fuzzy Technique, IEEE Transactions on Fuzzy Systems, June, 1995, Vol. 3, pp. 271-279.

[31] C. F. Juang and C. T. Lin. An On-line Self-constructing Neural Fuzzy Inference Network and its Applications. IEEE Transactions on Fuzzy Systems, Vol. 6, No. 1, 1998, pp.12-32.

[32] J. C. Bezdek, J. Keller, R. Krisnapuram, and N. R. Pal. Fuzzy Models and Algorithms for Pattern Recognition and Image Processing. Boston, MA: Kluwer, 1999.

[33] R. O. Duda, P. E. Hart, and D. G. Stork. Pattern Classification, 2nd ed. New York: Wiley, 2001.

[34] F. Höppner, F. Klawonn, R. Kruse, and T. Runkler. Fuzzy Cluster Analysis. New York: Wiley, 1999.

[35] S. Mitra and Y. Hayashi, Neuro-Fuzzy Rule Generation: Survey in Soft Computing Framework, IEEE Transactions on Neural Networks, Vol. 11, June 2000, pp. 748–768.

[36] N. K. Kasabov and Q. Song. DENFIS: Dynamic Evolving Neural-Fuzzy Inference System and its Application for Time-Series Prediction. IEEE Transactions on Fuzzy Systems, April, 2002, Vol. 10, pp. 144–154.

[37] J. Platt. A Resource Allocation Network for Function Interpolation, Neural Computation, 1991, Vol. 3, pp. 213–225.

[38] D. Deng and N. Kasabov, Evolving Self-Organizing Maps for Online Learning, Data Analysis and Modeling, in Proceedings IJCNN'2000 Neural Networks, Neural Computation: New Challenges Perspectives New Millennium, Vol.6, S.-I. Amari, C. L. Giles, M. Gori, and V. Piuri, Eds., New York, NY, 2000, pp. 3–8.

[39] N. Kasabov, Evolving Fuzzy Neural Networks—Algorithms, Applications and Biological Motivation in Methodologies for the Conception, Design and Application of Soft Computing, T. Yamakawa and G. Matsumoto, Eds, Singapore: World Scientific, 1998, pp. 271–274.

[40] J.E. Dennis and R. B. Schnabel, Numerical Methods for Unconstrained Optimization and Nonlinear Equations. Upper Saddle River, NJ: Prentice-hall, 1983.

[41] L. Ljung and T. Soderstrom, Theory and Practice of Recursive Identification. Cambridge, MA, MIT press, 1983.

[42] G.V. Puskorious and L.A. Feldkamp, Decoupled Extended Kalman Filter Training of feedforward Layered Networks, Neural Networks, IJCNN-91 Seattle International Joint Conference on Neural Networks, Vol. 1, 1991, pp. 771-777.

[43] S. Murtuza and S. F. Chorian, Node Decoupled Extended Kalman Filter Based Learning Algorithm for Neural Networks, International Symposium on Intelligent Control, August, 1994. Columbus, Ohio, USA, pp. 16-18.

[44] T. Takagi and M. Sugeno. Fuzzy Identification of system and its application to modeling and control. IEEE Transactions on Systems, Man and Cybernetics Part B, Vol. 15, 1985, pp. 116-132.

[45] M. Sugeno and M. Yasukawa, A Fuzzy Logic Based Approach to Qualitative Modeling, IEEE Transactions on Fuzzy Systems, February, 1993, Vol. 1, pp.7-31.

[46] S. Singhal and L. Wu, Training Multilayer Perceptrons with the Extended Kalman Algorithm. in Advances in Neural Information Processing Systems 1, Denver 1988, ed. D.S. Touretzky, Dan Mateo, CA: Morgan Kaufmann, pp133-140.

[47] Song, Q. and Kasabov. N., TWNFI- A Transductive Neuro-Fuzzy Inference System With Weighted Data Normalization for Personalized Modeling, Neural Networks, Dec. 2006, Vol.19, Issue 10, pp. 1591-1596.

[48] W. Wang and J. Vrbanek Jr., A Multi-Step Predictor for System State Forecasting, Measurement Science and Technology, 2007, Vol. 18, pp. 3673-3681.

[49] W. Wang and J. Vrbanek Jr., An Evolving Fuzzy Predictor for Industrial Applications, IEEE Transactions on Fuzzy Systems, TFS-2007-0204, (to be published in August 2008).

[50] W. Wang, An Intelligent System for Machinery Condition Monitoring, IEEE Transactions on Fuzzy Systems, 2008, Vol. 16, No. 1, pp. 110-122.

[51] W. Wang, An Enhanced Diagnostic System for Gear System Monitoring, IEEE Transactions on Systems, Man and Cybernetics, Part B, 2008, Vol. 34, pp. 102-112.

VITA

Mr. Vrbanek received his Electronic Engineering Technician and Electronic Engineering Technologist diplomas from RCC Institute of Technology in September 2002 and March 2003, respectively. He received his Bachelors in Electrical Engineering with a minor in Economics at Lakehead University in 2006.

Currently, Mr. Vrbanek is a graduate student and a researcher at Lakehead University. His specialization involves robotics and computational intelligence. He researches at the Laboratory for Intelligent Mechatronic Systems founded by Dr. Wilson Q. Wang. Also, he expects to receive a Master of Science in Control Engineering from Lakehead University in the fall of 2008. Mr. Vrbanek has been a Student Member of IEEE since 2001.